

Paper 172-2007

Using SAS/OR® to Determine the Optimal Path to Graduation

Sandra Archer, University of Central Florida, Orlando, FL

Robert L. Armacost, Higher Education Assessment and Planning Technologies,
Flagler Beach, FL

ABSTRACT

An integer program is formulated to create the programs of study for several student scenarios. SAS/OR is used to minimize the objective function of time to graduation subject to a set of assignment and policy constraints. The program is written with SAS 9.3.1 SP3 in Windows XP. The intended audience is familiar with Base SAS but new to the procedures available in SAS/OR.

INTRODUCTION

To assist students in their academic career planning at the University of Central Florida, many departments within the College of Engineering and Computer Science have developed a detailed schedule of course offerings that outlines which year and semester each class will be offered and also provides a generic flow-chart containing the path to graduation for a typical student. This flowchart does not address those students that enter the program with a variety of unique academic situations. A system is needed that will help students determine the fastest route to graduation (shortest number of consecutive semesters), while accounting for variables such as the desired number of credit hours per semester, required prerequisites, transfer-in credits, semesters preference (summer classes) and semester starting (students entering in the spring or summer). An additional preference may be considered for those degree requirements that allow for a selection among a set of elective courses. This system will then assist the student in determining which courses to register for each semester in order to graduate as soon as possible.

PROBLEM DEFINITION

There are numerous and diverse applications of optimization. Most applications involve the allocation of scarce resources to activities in order to obtain the best possible value for the overall performance measure. Optimization typically includes examples such as the maximization of profit or minimization of cost, generally with one objective function. When the objective function and the constraints are linear functions, the model is referred to as linear programming. Applications of linear programming include: production planning, personnel scheduling, cutting-stock/trim-loss problems, product-mix, transportation/transshipment, chemical blending and financial portfolio selection. In the course scheduling problem, the objective function is to minimize the number of consecutive semesters until graduation.

The general form of an optimization problem is comprised of:

- A set of decision variables – represent activities that the decision maker can control
- A set of constraints – restrictions on the decision variables, including resources and relationships
- Non-negativity constraints – decision variables must not be negative, when appropriate
- An objective function – a method for performance measurement for the entire system to be maximized or minimized while satisfying all constraints

The general assumptions of a linear program include:

- Linearity – the objective function and all constraints are linear functions
- Certainty – all coefficients of the objective function and constraints are known constants
- Divisibility – decision variables can have fractional values

In the case of integer programming, the divisibility constraint no longer holds. The formulation is similar to a linear program, with the additional constraint that each decision variable must be an integer. It is often necessary for the decision variables to be integer values when assigning people or machines to activities. Other examples include those cases when the decision variables are binary values of yes (1) or no (0). In the case of using integer programming for scheduling applications, the value of the decision variable is often a 1 or 0 to indicate whether an assignment of an activity to a particular timeslot has been made. To determine the optimal number of consecutive semesters from start to finish, the decision variable value of 1 will indicate if a course has been selected for a particular semester, zero otherwise. The problem is formulated as an integer program to create the programs of study for several student scenarios. Constraints are included (1) to ensure each course will be taken only once, (2) to restrict the number of courses per semester, (3) to ensure that all required courses will be assigned, (4) to allow for sets of selected electives, and (5) to order pre-requisite and successor courses. Finally, the course assignments are restricted to only those semesters as described in the five year course plan.

DEFINING A PARAMETERIZED PROBLEM WITH SAS

First, a set of indexes are established as:

$I \in \{1, 2, \dots, c\}$ = set of courses

$J \in \{1, 2, \dots, t\}$ = set of semesters

The decision variables are as follows:

Let $x_{ij} \in \{1, 0\}$ = 1 if course i is assigned to semester j ; 0 otherwise

Let $y_j \in \{1, 0\}$ = 1 if any course is assigned in semester j ; 0 otherwise

DATA AND PARAMETER PREPARATION

To approach this problem, two external data sources are needed. The first data set includes the expected timing of class offerings along with those class pre-requisites. A set of this data for graduate course offering projections for five years was provided by the department of Mechanical, Materials and Aerospace Engineering at the University of Central Florida. This information also included a list of up to three pre-requisite requirements per course. Each course is assigned a reference number and the semester in which they are offered is indicated with binary variables (one for each term). This "Five Year Course Plan" data are found in Appendix A. The second required data set includes the program requirements data from the graduate catalog for a Master of Science degree in Mechanical Engineering (M.S.M.E.). The example track selected is the Computer-Aided Mechanical Engineering program. These data were downloaded, formatted, and are shown in the "Program Requirements" data table in Appendix B. In this case, there are 4 required courses, a set of 7 track electives, and the remainder of allowable electives. The program requirement information resides in Excel and allows for a user to enter the following parameters in the highlighted cells: (1) total number of classes required, (2) the maximum allowable number of courses per term, and (3) the minimum number of track specialty courses. This is where the customization for each unique student's situation may be entered. If a course is already completed, it is simply deleted. If a program administrator chooses to replace a required course with another, the new course name and number may be entered.

FIVE YEAR COURSE PLAN DATA

The five year course plan displays 96 total courses with 15 semesters thus creating $96 \times 15 = 1,440$ course/semester combinations (x_{ij} 's), plus 15 semester indicators (y_i 's) for a total of 1,455 decision variables. To limit the processing requirements for this exercise, an assumption will be made that only those courses appearing in the list of "allowable courses" in the program requirements table will be scheduled. The five year course plan is imported into the SAS system by using Dynamic Data Exchange (DDE). Two text files are also generated to contain macro calls for later use in the code:

- "pre_req_macro.txt" contains a list of macro calls to macro "pre_req" for each prerequisite and successor class combination as seen here: "%pre_req(do i = 67; do k = 7);" where course number 67 (EML 5713) is a pre-requisite to course 7 (EAS 6135).
- "course_availability_macro.txt" contains a list of macro calls to macro "course_availability" for each course/semester offering combinations as shown here: "%course_availability(1,1);" where course 1 (EAS 4134) is expected to be offered in semester 1 (fall 2005).

PROGRAM REQUIREMENTS DATA

The program requirements data displayed in Appendix B has only 25 allowable courses, thereby requiring $25 \times 15 + 15 = 390$ decision variables. The global macro variable "t" is created to contain this number of semesters for later use. The example in Appendix B shows a user that has entered the following parameters: total number of classes required = 12; the maximum allowable number of courses per term = 4; and the minimum number of track specialty courses = 2. The program requirements data are imported into the SAS system, again using SAS DDE, and the following global macro variables are created:

- "required" = the count of required courses
- "required_list" = the list of course numbers of all required courses
- "track" = the count of track courses
- "track_list" = the list of course numbers of all track courses
- "allowable" = the count of all allowable courses
- "allowable_classes_list" = the list of all allowable courses
- "number_nreq" = parameter entered by the user for the number of courses the student must take
- "number_ntrack" = parameter entered by the user for the number of track specialty classes required
- "number_nmax" = parameter entered by the user for the maximum number of courses the student may enroll in any semester

BUILDING MPS DATA IN A SPARSE DATA FORMAT

To solve this integer program with SAS/OR, data are required to be entered in the MPS format, an input format that is common to several linear programming software packages. The MPS format essentially contains a row for the objective function and every constraint, and a column for every decision variable (as well as a column to identify the rows, describe the row type, and describe the right hand side of each equation). The values contained in the data set describe the coefficients of each term. In the example we are using here, the number of columns is 390 (decision

variables) plus 3 (id, type and RHS). Because the system allows users to enter different numbers of allowable courses, the number of decision variables will vary. In order to accommodate various numbers of decision variables, we use instead the "sparse" MPS format that is created dynamically for every system run. The macro variables described above can be seen in the example code below to control the number of rows entered into the data set. The sparse MPS format has a fixed number of columns with each row describing a single element of the traditional MPS format, which will vary based on user input. A data set that contains the sparse input must have a column to describe the type, row, column, and coefficient for each term. The "type" variable describes if the objective function is a MIN or MAX and if the constraints are EQ, LE, GE, etc. The combination of each "row" and "column" variable describe the elements of the traditional MPS format. The coefficients values for the objective function and constraints are contained in the "coefficient" variable. This variable will also contain any lower and upper bounds, and identify model variables with that are BASIC, FIXED, BINARY, or INTEGER. The sparse MPS data is constructed in a data set called "model" by the code found below along with an explanation of the portion of the linear program.

OBJECTIVE FUNCTION

The objective function is to minimize the number of consecutive semesters until the program of study requirements are completed. The first term of the objective function is the sum of the binary y indicators (0 or 1 for each semester without or with enrollment) multiplied by a weight that incrementally increases for each future semester. These weights are increasing so the objective function will prefer to assign courses to terms in earlier semesters. This term will also reduce the number of semesters in which a student is enrolled by encouraging the grouping of courses into the least number of semesters. The second term of the objective function is the sum of indicators of assigned courses. This is included in the objective function simply to discourage assigning excess courses. Please note, the formulation is written below two ways for clarity: first the objective functions and constraints are written in a long form and secondly with indices for a short-hand version. The long form also serves as an example case.

<p>Minimize time to degree D =</p> $1y_1 + 2y_2 + \dots + ty_t$ $+ x_{11} + x_{12} + \dots + x_{ij}$ <p>Constraint: Integer (binary) constraints on the decision variables: $x_{ij} \in \{1,0\}$ and $y_i \in \{1,0\}$</p>	$\min \sum_{j=1}^t w_j y_j + \sum_{i=1}^c \sum_{j=1}^t x_{ij}$ <p>$j = 1,2,\dots,t; w_j = 1,2,\dots,t; i = 1, 2, \dots, c$</p>																														
<p>Code description:</p> <p>The below code begins the DATA STEP to create the data set "model" with four columns: <code>_type_</code> <code>_row_</code> <code>_col_</code> <code>_coef_</code> and adds a set of rows similar to those shown here. The first row indicates that the objective function named "time_to_degree" should be minimized. There are 375 rows added similar to the second row for every course/semester combination decision variable, (the one displayed is for course 44, semester 1) and 15 rows added similar to the third row for every semester decision variable (the row displayed is for semester 1 only). The coefficient values in the last column correspond to the coefficients as seen in the objective function displayed above. Similarly, there are 375 +15 rows are added to indicate that all decision variables are binary.</p>	<p>Sparse MPS format:</p> <table border="1"> <thead> <tr> <th></th> <th><code>_type_</code></th> <th><code>_row_</code></th> <th><code>_col_</code></th> <th><code>_coef_</code></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>min</td> <td>time_to_degree</td> <td></td> <td></td> </tr> <tr> <td>2</td> <td></td> <td>time_to_degree</td> <td>x44_1</td> <td>1</td> </tr> <tr> <td>3</td> <td></td> <td>time_to_degree</td> <td>y1</td> <td>1</td> </tr> <tr> <td>4</td> <td>binary</td> <td>binary_constraint</td> <td>x44_1</td> <td>1</td> </tr> <tr> <td>5</td> <td>binary</td> <td>binary_constraint</td> <td>y1</td> <td>2</td> </tr> </tbody> </table> <p>391 rows added to data set "model"</p>		<code>_type_</code>	<code>_row_</code>	<code>_col_</code>	<code>_coef_</code>	1	min	time_to_degree			2		time_to_degree	x44_1	1	3		time_to_degree	y1	1	4	binary	binary_constraint	x44_1	1	5	binary	binary_constraint	y1	2
	<code>_type_</code>	<code>_row_</code>	<code>_col_</code>	<code>_coef_</code>																											
1	min	time_to_degree																													
2		time_to_degree	x44_1	1																											
3		time_to_degree	y1	1																											
4	binary	binary_constraint	x44_1	1																											
5	binary	binary_constraint	y1	2																											

```

data model;
  format _type_ $20. _row_ $20. _col_ $20. _coef_ 10.0;
  keep _type_ _row_ _col_ _coef_;
  /* build the object function and binary constraint*/
  if _n_=1 then do;
    _type_='min'; _row_='time_to_degree'; _col_=''; _coef_='.'; output;
  end;
  do i = %allowable_classes_list 0; if i > 0 then do;
    do j = 1 to &t.;
      _type_=''; _row_='time_to_degree';
      _col_=compress('x' || put(i,2.) || '_' || put(j,2.)); _coef_= 1; output;
      _type_='binary'; _row_='binary_constraint';
      _col_=compress('x' || put(i,2.) || '_' || put(j,2.)); _coef_= 1; output;
    end;
  end;end;
  do j = 1 to &t.;
    _type_=''; _row_='time_to_degree'; _col_=compress('y' || put(j,2.));
    _coef_= put(j,2.); output;
    _type_='binary'; _row_='binary_constraint';
    _col_=compress('y' || put(j,2.)); _coef_= 2; output;
  end;

```

CONSTRAINT SET A: SEMESTER ASSIGNMENT

This constraint set uses the “big M” method to assign a binary value of 1 to y_j if a course is assigned to semester j .

<p>Constraint A: Create y_j indicator if semester j has a course</p> $\begin{array}{rcllcl} j=1 & x_{11}+ & x_{21}+ & \dots & + x_{c1} & \leq 1000 y_1 \\ j=2 & x_{12}+ & x_{22}+ & \dots & + x_{c2} & \leq 1000 y_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ j=t & x_{1t}+ & x_{2t}+ & \dots & + x_{ct} & \leq 1000 y_t \end{array}$	$\sum_{i=1}^c x_{ij} \leq M y_j \quad \forall j$																				
<p>Code description: The below code will add the rows needed for the sparse MPS format for constraint set A, as seen in the table to the right. The code adds 375 rows similar to the sixth row, indicating that all 375 decision variables representing course/semester combinations have a coefficient of 1 and is a less than or equal to constraint. 15 rows are added similar to the seventh row shown here, indicating that the coefficient of the y_j terms is -1000, and 15 rows are added similar to the eighth row indicating that the RHS of each inequality is zero, similar to the fourth and fifth row shown here.</p>	<p>Sparse MPS format:</p> <table border="1"> <thead> <tr> <th></th> <th><u>_type_</u></th> <th><u>_row_</u></th> <th><u>_col_</u></th> <th><u>_coef_</u></th> </tr> </thead> <tbody> <tr> <td>6</td> <td>le</td> <td>C_a1</td> <td>x44_1</td> <td>1</td> </tr> <tr> <td>7</td> <td>le</td> <td>C_a1</td> <td>y1</td> <td>-1000</td> </tr> <tr> <td>8</td> <td>le</td> <td>C_a1</td> <td>_RHS_</td> <td>0</td> </tr> </tbody> </table> <p>405 rows added to data set “model”</p>		<u>_type_</u>	<u>_row_</u>	<u>_col_</u>	<u>_coef_</u>	6	le	C_a1	x44_1	1	7	le	C_a1	y1	-1000	8	le	C_a1	_RHS_	0
	<u>_type_</u>	<u>_row_</u>	<u>_col_</u>	<u>_coef_</u>																	
6	le	C_a1	x44_1	1																	
7	le	C_a1	y1	-1000																	
8	le	C_a1	_RHS_	0																	

```
/* constraint set A: create a y_j indicator if semester j has a course */
do j = 1 to &t.;
  do i = %allowable_classes_list 0; if i > 0 then do;
    _type_='le'; _row_=compress('C_a' || put(j,2.)); _col_=compress('x' || put(i,2.) ||
      '_' || put(j,2.)); _coef_= 1; output;
  end;end;
  _type_='le'; _row_=compress('C_a' || put(j,2.)); _col_=compress('y' || put(j,2.));
  _coef_= -1000; output;
  _type_='le'; _row_=compress('C_a' || put(j,2.)); _col_='_RHS_'; _coef_= 0; output;
end;
```

CONSTRAINT SET B: COURSE NON-REPETITION

This constraint set ensures that each course will be taken only once.

<p>Constraint B: Assign a class only once; sum over t semester for all classes</p> $\begin{array}{rcllcl} i=1 & x_{11}+ & x_{12}+ & \dots & + x_{1t} & \leq 1 \\ i=2 & x_{21}+ & x_{22}+ & \dots & + x_{2t} & \leq 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ i=c & x_{c1}+ & x_{c2}+ & \dots & + x_{ct} & \leq 1 \end{array}$	$\sum_{j=1}^t x_{ij} \leq 1 \quad \forall i$															
<p>Code description: The below code will add 375 rows to the data set model similar to the ninth row shown here (again, only course 44 semester 1 combination is shown). These rows indicate that the constraints are “less than or equal to” and the coefficients of each term is 1. There are 25 rows similar to the tenth row shown here to indicate that the RHS of the constraints are 1.</p>	<p>Sparse MPS format in data set “model”:</p> <table border="1"> <thead> <tr> <th></th> <th><u>_type_</u></th> <th><u>_row_</u></th> <th><u>_col_</u></th> <th><u>_coef_</u></th> </tr> </thead> <tbody> <tr> <td>9</td> <td>le</td> <td>C_b44</td> <td>x44_1</td> <td>1</td> </tr> <tr> <td>10</td> <td>le</td> <td>C_b44</td> <td>_RHS_</td> <td>1</td> </tr> </tbody> </table> <p>400 rows added to data set “model”</p>		<u>_type_</u>	<u>_row_</u>	<u>_col_</u>	<u>_coef_</u>	9	le	C_b44	x44_1	1	10	le	C_b44	_RHS_	1
	<u>_type_</u>	<u>_row_</u>	<u>_col_</u>	<u>_coef_</u>												
9	le	C_b44	x44_1	1												
10	le	C_b44	_RHS_	1												

```
/* constraint set B: Assign a class only once; sum semesters for all classes */
do i = %allowable_classes_list 0; if i > 0 then do;
  do j = 1 to &t.;
    _type_='le'; _row_=compress('C_b' || put(i,2.));
    _col_=compress('x' || put(i,2.) || '_' || put(j,2.)); _coef_= 1; output;
  end;
  _type_='le'; _row_=compress('C_b' || put(i,2.)); _col_='_RHS_'; _coef_= 1; output;
end;end;
```

CONSTRAINT SET C: COURSES PER SEMESTER LIMIT

This constraint set ensures that a student will take an appropriate number of courses each semester. This example shows that the maximum number of courses per semester is four and may be adjusted based on the case. For example, the number may be increased for full-time students or decreased for part-time students, and summer terms may be assigned to zero. The inequality symbol may be changed to an equality constraint for those students that require a specific number of courses per term due to scholarship or financial aid requirements.

<p>Constraint C: Assign at most 4 courses per term; Sum of courses assigned per term ≤ 4</p> $\begin{array}{rcccccc} j = 1 & x_{11} + & x_{21} + & \dots & + x_{c1} & \leq 4 \\ j = 2 & x_{12} + & x_{22} + & \dots & + x_{c2} & \leq 4 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ j = t & x_{1t} + & x_{2t} + & \dots & + x_{ct} & \leq 4 \end{array}$	$\sum_{i=1}^c x_{ij} \leq n \quad \forall j$ <p>where n is the maximum number of courses to assign to any semester</p>															
<p>Code description: The below code will add 375 rows similar to the eleventh one as shown here, indicating that the terms above have a coefficient of 1 and the constraints are less than or equal to. There are 15 additional rows added similar to the twelfth row here to indicate that the RHS of the constraints is 4.</p>	<p>Sparse MPS format in data set "model":</p> <table border="1"> <thead> <tr> <th></th> <th>_type_</th> <th>_row_</th> <th>_col_</th> <th>_coef_</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>le</td> <td>C_c1</td> <td>x44_1</td> <td>1</td> </tr> <tr> <td>12</td> <td>le</td> <td>C_c1</td> <td>_RHS_</td> <td>4</td> </tr> </tbody> </table> <p>390 rows added to data set "model"</p>		_type_	_row_	_col_	_coef_	11	le	C_c1	x44_1	1	12	le	C_c1	_RHS_	4
	type	_row_	_col_	_coef_												
11	le	C_c1	x44_1	1												
12	le	C_c1	_RHS_	4												

```
/* constraint set C: At most 4 courses per term; Sum courses per semester <= 4 */
do j = 1 to &t.;
  do i = %allowable_classes_list 0; if i > 0 then do;
    _type_='le'; _row_=compress('C_c' || put(j,2.));
    _col_=compress('x' || put(i,2.) || '_' || put(j,2.)); _coef_= 1; output;
  end;end;
  _type_='le'; _row_=compress('C_c' || put(j,2.)); _col_='_RHS_';
  _coef_= &number_nmax.; output;
end;
```

CONSTRAINT SET D: REQUIRED COURSE ASSIGNMENTS

This constraint set will use the program requirements table to determine the list of required classes for a particular degree program and ensures that all of these courses will be assigned. This may also be used for courses that the student expresses a specific interest in.

<p>Constraint D: Required courses must be taken; Assume courses 44, 49, 54, 72 are required</p> $\begin{array}{rcccccc} x_{44_1} + & x_{44_2} + & \dots & + x_{44_t} & = & 1 \\ x_{49_1} + & x_{49_2} + & \dots & + x_{49_t} & = & 1 \\ x_{54_1} + & x_{54_2} + & \dots & + x_{54_t} & = & 1 \\ x_{72_1} + & x_{72_2} + & \dots & + x_{72_t} & = & 1 \end{array}$	$\sum_{j=1}^t x_{rj} = 1 \quad \forall r \in R$ <p>where R is the set of required courses or requested elective</p>															
<p>Code description: The below code will add 375 rows similar to the thirteenth row shown here that will indicate that the decision variables have a coefficient of 1 and the constraints are equalities. There are an additional 4 rows added, similar to the fourteenth row shown here, indicating that the RHS of these constraints is 1.</p>	<p>Sparse MPS format in data set "model":</p> <table border="1"> <thead> <tr> <th></th> <th>_type_</th> <th>_row_</th> <th>_col_</th> <th>_coef_</th> </tr> </thead> <tbody> <tr> <td>13</td> <td>eq</td> <td>C_d44</td> <td>x44_1</td> <td>1</td> </tr> <tr> <td>14</td> <td>eq</td> <td>C_d44</td> <td>_RHS_</td> <td>1</td> </tr> </tbody> </table> <p>379 rows added to data set "model"</p>		_type_	_row_	_col_	_coef_	13	eq	C_d44	x44_1	1	14	eq	C_d44	_RHS_	1
	type	_row_	_col_	_coef_												
13	eq	C_d44	x44_1	1												
14	eq	C_d44	_RHS_	1												

```
/* constraint set D: Required courses must be taken; Assume courses 44, 49, 54, 72
are required */
do i = %required_list 0; if i > 0 then do;
  do j = 1 to &t.;
    _type_='eq'; _row_=compress('C_d' || put(i,2.));
    _col_=compress('x' || put(i,2.) || '_' || put(j,2.)); _coef_= 1; output;
  end;
  _type_='eq'; _row_=compress('C_d' || put(i,2.)); _col_='_RHS_'; _coef_= 1; output;
end;end;
```

CONSTRAINT SET E: ELECTIVE COURSE ASSIGNMENTS

This set of constraints allows for sets of electives to select from. The example below shows how the constraint would be written for a degree program that requires at least 2 of a set of 7 electives. An additional constraint written in this format will also be used to set total number of courses required for a degree program (for example, $k = 12$ and N is the set of all allowable courses).

<p>Constraint E1: Elective classes must be selected; Assume at least 2 of these 7 courses are required: 52, 43, 60, 71, 90, 89, 93.</p> $ \begin{array}{cccccc} X_{52_1} + & X_{52_2} + & \dots & + X_{52_t} & + & \\ X_{43_1} + & X_{43_2} + & \dots & + X_{43_t} & + & \\ \dots & \dots & \dots & \dots & + & \\ X_{93_1} + & X_{93_2} + & \dots & + X_{93_t} & & \geq 2 \end{array} $	$ \sum_{i \in N} \sum_{j=1}^t x_{ij} \geq k $ <p>where one must select at least k of n courses from course set N</p>																									
<p>Code description: The below code will add 105 rows similar to the fifteenth row shown here for every track course/semester combination and 1 row shown in the sixteenth row here to indicate the RHS of the constraint equation is 2. In a very similar fashion, 375 rows for each decision variable are added (similar to the seventeenth row here), and 1 row as shown in the eighteenth row to indicate that the sum of all decision variables should be equal to 12.</p>	<p>Sparse MPS format in data set "model":</p> <table border="1" data-bbox="938 594 1438 737"> <thead> <tr> <th></th> <th><u>_type_</u></th> <th><u>_row_</u></th> <th><u>_col_</u></th> <th><u>_coef_</u></th> </tr> </thead> <tbody> <tr> <td>15</td> <td>ge</td> <td>C_e1</td> <td>x52_1</td> <td>1</td> </tr> <tr> <td>16</td> <td>ge</td> <td>C_e1</td> <td>_RHS_</td> <td>2</td> </tr> <tr> <td>17</td> <td>eq</td> <td>C_e2</td> <td>x44_1</td> <td>1</td> </tr> <tr> <td>18</td> <td>eq</td> <td>C_e2</td> <td>_RHS_</td> <td>12</td> </tr> </tbody> </table> <p>106 + 376 rows added to data set "model"</p>		<u>_type_</u>	<u>_row_</u>	<u>_col_</u>	<u>_coef_</u>	15	ge	C_e1	x52_1	1	16	ge	C_e1	_RHS_	2	17	eq	C_e2	x44_1	1	18	eq	C_e2	_RHS_	12
	<u>_type_</u>	<u>_row_</u>	<u>_col_</u>	<u>_coef_</u>																						
15	ge	C_e1	x52_1	1																						
16	ge	C_e1	_RHS_	2																						
17	eq	C_e2	x44_1	1																						
18	eq	C_e2	_RHS_	12																						

```

/* constraint set E1: Track elective classes must be selected */
do i = %track_list 0; if i > 0 then do;
  do j = 1 to &t.;
    _type_='ge';_row_=compress('C_e1');
    _col_=compress('x'||put(i,2.)||'_'||put(j,2.)); _coef_= 1; output;
  end;
end; end;
_type_='ge'; _row_=compress('C_e1'); _col_='_RHS_'; _coef_= &number_ntrack.; output;

/* constraint set E2: Total number of required classes is 12 */
do i = %allowable_classes_list 0; if i > 0 then do;
  do j = 1 to &t.;
    _type_='eq';_row_=compress('C_e2');
    _col_=compress('x'||put(i,2.)||'_'||put(j,2.)); _coef_= 1; output;
  end;
end; end;
_type_='eq'; _row_=compress('C_e2'); _col_='_RHS_'; _coef_= &number_nreq.; output;

```

CONSTRAINT SET F: PRE-REQUISITES AND COURSE ORDERING

This set of constraints provides for the ordering of pre-requisite courses. The example below demonstrates the constraint set if course 3 is a pre-requisite for course 5. An additional set of constraints would be added for each additional pre-requisite pairing. This current formulation will not add additional courses to accommodate pre-requisites; only order those that have been selected.

<p>Constraint F: Pre-requisite classes must be completed in an earlier semester; assume course 3 is a pre-requisite for course 5</p> $\begin{array}{rcll} X_{51} & & - X_{31} & < 0 \\ X_{52} & & - X_{31} & \leq 0 \\ X_{53} & & - X_{32} & - X_{31} \leq 0 \\ X_{54} & - X_{33} & - X_{32} & - X_{31} \leq 0 \\ X_{55} & - X_{34} & - X_{33} & - X_{32} - X_{31} \leq 0 \\ \dots & & & \\ X_{5t} & - X_{3\ t-1} & \dots & - X_{32} - X_{31} \leq 0 \end{array}$	$x_{a1} < x_{b1} \text{ and}$ $x_{an} \leq \sum_{i=1}^{n-1} x_{bi}$ <p>where course b is a pre-requisite for course a for all semesters $n \in \{2,3,\dots, t\}$</p>																				
<p>Code description: The below code will add 2,226 rows to the model data set, similar to the nineteenth row shown here since one row like this is needed for every negative term as shown above. Additionally, 315 rows similar to the twentieth and twenty-first shown here will be added for every pre-requisite pairing that occurs.</p>	<p>Sparse MPS format in data set "model":</p> <table border="1"> <thead> <tr> <th></th> <th>type</th> <th>row</th> <th>col</th> <th>coef</th> </tr> </thead> <tbody> <tr> <td>19</td> <td>le</td> <td>C_f_67_7_1</td> <td>x67_1</td> <td>-1</td> </tr> <tr> <td>20</td> <td>le</td> <td>C_f_67_7_1</td> <td>x7_1</td> <td>1</td> </tr> <tr> <td>21</td> <td>le</td> <td>C_f_67_7_1</td> <td>_RHS_</td> <td>0</td> </tr> </tbody> </table> <p>2,541 rows added to data set "model"</p>		type	row	col	coef	19	le	C_f_67_7_1	x67_1	-1	20	le	C_f_67_7_1	x7_1	1	21	le	C_f_67_7_1	_RHS_	0
	type	row	col	coef																	
19	le	C_f_67_7_1	x67_1	-1																	
20	le	C_f_67_7_1	x7_1	1																	
21	le	C_f_67_7_1	_RHS_	0																	

```

/* constraint set F: Pre-requisites must be completed in an earlier semester */
/* as a reminder, the macro calls appear as: %pre_req(do i = 67; do k = 7); */
%macro pre_req(p_is_a_pre_req_for_q);
do j = 1;
  &p_is_a_pre_req_for_q.;
  _type_='le';
  _row_=compress('C_f' || '_' || put(i,2.) || '_' || put(k,2.) || '_' || put(j,2.));
  _col_=compress('x' || put(i,2.) || '_' || put(j,2.)); _coef_ = -1; output;
  _type_='le';
  _row_=compress('C_f' || '_' || put(i,2.) || '_' || put(k,2.) || '_' || put(j,2.));
  _col_=compress('x' || put(k,2.) || '_' || put(j,2.)); _coef_ = 1; output;
end; end;
  _type_='le';
  _row_=compress('C_f' || '_' || put(i,2.) || '_' || put(k,2.) || '_' || put(j,2.));
  _col_='_RHS_'; coef_ = 0; output;
end;
do j = 2 to &t.;
  &p_is_a_pre_req_for_q.;
  do countback = 1 to &t.; if j-countback > 0 then do;
    _type_='le';
    _row_=compress('C_f' || '_' || put(i,2.) || '_' || put(k,2.) || '_' ||
      put(j,2.));
    _col_=compress('x' || put(i,2.) || '_' || put(j-countback,2.));
    _coef_ = -1; output;
  end; end;
  _type_='le';
  _row_=compress('C_f' || '_' || put(i,2.) || '_' || put(k,2.) || '_' || put(j,2.));
  _col_=compress('x' || put(k,2.) || '_' || put(j,2.)); _coef_ = 1; output;
end; end;
  _type_='le';
  _row_=compress('C_f' || '_' || put(i,2.) || '_' || put(k,2.) || '_' || put(j,2.));
  _col_='_RHS_'; _coef_ = 0; output;
end;
%mend;
%inc 'C:\pre_req_macro.txt'; *contains the macro calls;

```

CONSTRAINT SET G: COMPLIANCE WITH PLANNED COURSE OFFERING

This constraint set uses the five year course plan data to force those decision variables to zero where a course is not offered in that semester.

<p>Constraint G: Do not assign a course to a semester if it is unavailable; Assume class a is not offered in semester b</p> $x_{ab} = 0$	$x_{ab} = 0 \quad x_{ab} \notin I(j) \quad \text{where } I(j) = \text{set of courses available during semester } j$															
<p>Code description: The below code will add 250 rows similar to the twenty-second row shown here to indicate all those semesters where courses are unavailable. (Here, course 7 is not available in semester 1). One additional row, shown in the twenty-third shown here, is added to indicate that the sum of all these decision variables should be zero.</p>	<p>Sparse MPS format in data set "model":</p> <table border="1" data-bbox="938 443 1433 533"> <thead> <tr> <th></th> <th><u>_type_</u></th> <th><u>_row_</u></th> <th><u>_col_</u></th> <th><u>_coef_</u></th> </tr> </thead> <tbody> <tr> <td>22</td> <td>eq</td> <td>C_g</td> <td>x7_1</td> <td>1</td> </tr> <tr> <td>23</td> <td>eq</td> <td>C_g</td> <td>_RHS_</td> <td>0</td> </tr> </tbody> </table> <p>251 rows added to data set "model"</p>		<u>_type_</u>	<u>_row_</u>	<u>_col_</u>	<u>_coef_</u>	22	eq	C_g	x7_1	1	23	eq	C_g	_RHS_	0
	<u>_type_</u>	<u>_row_</u>	<u>_col_</u>	<u>_coef_</u>												
22	eq	C_g	x7_1	1												
23	eq	C_g	_RHS_	0												

```

/* constraint set G: Do not assign a course to a semester if it is unavailable */
/* as a reminder, the macro calls appear as: %course_availability(1,1); */

%macro course_availability(a,b);
  _type_='eq'; _row_=compress('C_g');
  _col_=compress('x'|put(&a.,2.)|'|_'|put(&b.,2.)); _coef_= 1; output;
%mend;
  _type_='eq'; _row_=compress('C_g'); _col_='_RHS_'; _coef_= 0; output;
%inc 'C:\course_availability_macro.txt'; *contains the macro calls;
run;

```

Finally, the "run" statement above finishes the DATA STEP that creates the SAS data set "model". The data set contains a total of 5,629 rows.

SOLVING THE LP

The final piece of SAS code feeds the SAS data that is in the sparse MPS format into the SAS system procedure PROC LP. Several options are set to increase the default number of allowable iterations as well as processing time. Finally, the primal solution is output to an Excel workbook, where the results are linked back to the original user interface sheet.

```

/* Solve the LP */
proc lp
data = model sparsedata
primalout=work.primal
dualout= dual
activeout = work.active

MAXIT1= 1000000
MAXIT2= 1000000
imaxit= 10000000
time= 100000
PRINTFREQ= 10000
; run;

PROC EXPORT DATA= WORK.primal
OUTFILE= "C:\primal_solution.xls"
DBMS=EXCEL REPLACE;
SHEET="Solution";
RUN;

```

The output data set "primal" contains a row for each decision variable, constraint, and the objective function with the following columns:

- _VAR_ = contains the name of the decision variable or constraint name
- _TYPE_ = indicates the input data type (BINARY, SLACK, SURPLUS, or OBJECT)

- `_STATUS_` = indicates the decision variable or constraint's status (`_ALTER_`, `_BASIC_`, `_DEGEN_`, or `_UPPER_`). In this case, decision variables with a status of "ALTER" indicate the existence of alternative optimal solutions for this student scenario.
- `_LBOUND_` = contains the lower bound of the variables needed to arrive at the integer solution (0 or 1). In this case, `y1`, `y3`, `y5`, `y8` variables have a lower bound of 1
- `_VALUE_` = contains the value of the decision variable (0 or 1), constraint or objective function for the solution provided. In this case, the value of the objective function is 29. This number is meaningless to the user, since it is the sum of semester numbers multiplied by a set of weights.
- `_UBOUND_` = contains the upper bound of the variables needed to arrive at the integer solution (0 or 1). In this case, the `y2` variable has an upper bound of 0
- `_PRICE_` = contains the "price" coefficient of each decision variable in the objective function. In this case, the "price" are the weights (from 1 to 15) applied to the `y` indicators for each semester in the objective function.
- `_R_COST_` = contains the value of the reduced cost for the solution

The end results received by the user is the "solution semester" column in the program requirements data, indicating the suggested term number that the student may register for these classes. The data have been extracted and are displayed in Tables 1 and 2 for two different scenarios. After satisfying all constraints, the student is recommended to register in semesters 1, 3, 5, and 8 (FA05, SU06, SP07, and SP08) for the courses shown in Table 1. This wide timeframe spread is due to the requirements imposed by the ordering of pre-requisite classes.

Table 1. Optimal Course Schedule considering ordering of pre-requisite requirements

	Fall 05	Spring 06	Sum 06	Fall 06	Spring 07	Sum 07	Fall 07	Spring 08
EML 5060			EML 5713		EML 5271			EML 6067
EML 5211					EML 5237			EAS 6138
EML 5402					EML 5532			EAS 6185
EML 6971								EML 6085

Without the pre-requisite ordering constraint, the student is able to complete all 12 required classes in semesters 1, 2, and 3 (FA05, SP 06, and SU06) as shown in Table 2.

Table 2. Optimal Course Schedule ignoring pre-requisite ordering

	Fall 05	Spring 06	Sum 06	Fall 06	Spring 07	Sum 07	Fall 07	Spring 08
EML 5060		EML 5271	EML 5025					
EML 5211		EML 6067	EML 5532					
EML 6547		EML 6725	EML 5713					
EML 6712		EML 5131	EML 6971					

CONCLUSION

The above exercise demonstrates that solving integer linear programs is a reasonable method for generating plans of study for graduate students, and the same method may be used to develop programs of study for undergraduate students. Such a system requires perfect information regarding the semesters in which courses will be offered in the next several years. Because students have different academic histories and program of study needs, a user interface to customize program requirement input would be ideal. The example here demonstrates that the inclusion of a set of constraints requiring the ordering of prerequisite courses can drastically increase the time to graduation. Therefore, the ability to relax some of these pre-requisite restrictions may improve the solutions. This model could be used by the department in a comprehensive study of pre-requisite requirements and would illustrate pre-requisite "bottlenecks." This could lead to a review of pre-requisite requirements and may result in designating co-requisites or simply rescheduling course offerings. Classes that have academic standing requirements before registration are not currently considered and would require a much more complex system involving individual student records. Many assumptions are made within this application including: students may register for any courses during any term with no restrictions and no scheduling conflicts within the semester, and students will earn passing grades.

This paper demonstrates that SAS is a tool that can accept parameters and generate customized linear program MPS data for solving with SAS/OR procedures. Further extensions of this system would increase the flexibility of the user input interface, such as the ability for students to enter preferences for sets of electives over others. Also, a more user-friendly interface may check the parameters and prompt users for corrections before MPS data generation. Adding flexibility to the user interface and enhancing the dynamic generation of the sparse MPS code will increase the unique student situations that can be served. Additional extensions include other advantages for end users such as printing out several optional programs of study. Because there may be more than one optimal solution, as demonstrated by the existence of `_ALTER_` flags in the "STATUS" variable in the output data set in the example

above, there is an opportunity to provide students with several possible optimal programs of study from which to select.

REFERENCES

- Bunn, Derek, *Analysis for Optimal Decisions*, John Wiley & Sons, NY, 1982
 Daskalaki, S and Birbas, T, *Efficient Solutions for a university timetabling problem through integer programming*, European Journal of Operational Research 160 (2005) 106-120
 Hillier, Frederick and Lieberman, Gerald, *Introduction to Operations Research*, McGraw-Hill, 1995
 Hopkins, David S. P. and Massy, William F., *Planning Models for Colleges and Universities*, Stanford University Press, Stanford, CA, 1981
 Martin, Clarence, *Ohio University's College of Business Uses Integer Programming to Schedule Classes*; Interfaces, Vol 34, No.6, Nov-Dec 2004, pp. 460-465
 Murty, Katta, *Network Programming*, Prentice-Hall, NJ, 1992
 SAS/OR® 9.1.3 User's Guide: *Mathematical Programming 2.1*, SAS Institute Inc., Cary, NC, USA, 2006
 Wu, Nesa and Coppins, Richard, *Linear Programming*, McGraw-Hill, 1981

CONTACT INFORMATION

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

Ms. Sandra Archer
 University Analysis and Planning Support
 University of Central Florida
 12424 Research Parkway, Suite 215
 Orlando, FL 32826-3207
 407-882-0287
 archer@mail.ucf.edu
 http://uaps.ucf.edu

Dr. Robert L. Armacost
 Higher Education Assessment and Planning Technologies
 P.O. Box 2237
 Flagler Beach, FL 32136
 386-439-7486
 armacost@mail.ucf.edu

Sandra Archer is Interim Director for the office of University Analysis and Planning Support at the University of Central Florida with primary responsibility for the design and development of knowledge management systems to support University operations. This includes developing enrollment planning models, supporting university strategic planning and benchmarking efforts, and conducting special studies to support program management and planning decisions. Sandra has a MS in Statistical Computing and is a PhD student in Industrial Engineering.

Dr. Robert L. Armacost is retired from the University of Central Florida where he was Director of University Analysis and Planning Support and an Associate Professor in the Industrial Engineering and Management Systems Department. He was an Associate Professor Marquette University and served over 20 years in the U.S. Coast Guard. His graduate degrees are in Operations Research. He has extensive research and consulting experience in planning and optimization of business operations.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
 Other brand and product names are trademarks of their respective companies.

APPENDIX B: PROGRAM REQUIREMENTS TABLE

Program Requirements Table for Degree Program: MS in ME Computer-Aided Track

http://www.graduate.ucf.edu/currentGradCatalog/content/Degrees/ACAD_PROG_118.cfm

Master of Science in Mechanical Engineering		
Computer-Aided Mechanical Engineering Track		
<i>Enter Total Classes Required:</i>	12	
<i>Enter Max classes per term:</i>	4	
	Course Number	Solution Semester
Required Courses:		
EML 5060 Mathematical Methods in Mechanical, Materials and	44	1
EML 5211 Continuum Mechanics (3 credit hours)	49	1
EML 5271 Intermediate Dynamics (3 credit hours)	54	5
EML 6067 Finite Elements in Mechanical, Materials and Aeros	72	8
<i>Enter # of courses from track specialty courses:</i>	2	
EML 5237 Intermediate Mechanics of Materials (3 credit hours)	52	5
EML 5025C Engineering Design Practice (3 credit hours)	43	-
EML 5532C Computer-Aided Design for Manufacture (3 credit h	60	5
EML 6062 Boundary Element Methods in Engineering (3 credit	71	-
EML 6547 Engineering Fracture Mechanics in Design (3 credit	90	-
EML 6305C Experimental Mechanics (3 credit hours)	89	-
EML 6725 Computational Fluid Dynamics and Heat Transfer I	93	-
Electives		
EAS 6138 Advanced Gas Dynamics (3 credit hours)	7	8
EAS 6185 Turbulent Flow (3 credit hours)	8	8
EML 5105 Gas Kinetics and Statistical Thermodynamics (3 cr	46	-
EML 5402 Turbomachinery (3 credit hours)	59	1
EML 6155 Convection Heat Transfer (3 credit hours)	79	-
EML 6712 Mechanics of Viscous Flow (3 credit hours)	92	-
EML 5066 Computational Methods in Mechanical, Materials ar	45	-
EML 5131 Combustion Phenomena (3 credit hours)	47	-
EML 5152 Intermediate Heat Transfer (3 credit hours)	48	-
EML 5713 Intermediate Fluid Mechanics (3 credit hours)	67	3
EML 6154 Conduction Heat Transfer (3 credit hours)	78	-
EML 5546 Engineering Design with Composite Materials (3 cr	61	-
EML 6971 Thesis (6 credit hours)	96	1
EML 6085 Research Methods in MMAE (required for nonthesis	74	8
		-
		-