

Getting Started with Optimization Modeling at Penn State

(Under Preparation)
Last Updated: January 21, 2008

Anshuman Gupta
Research Computing and Cyberinfrastructure (RCC),
Information Technology Services (ITS)
The Pennsylvania State University, University Park, PA 16802

Contact Information: 222B Computer Bldg, University Park, PA 16802
Ph: 814-865-6081, Email: axg218@psu.edu

Optimization Software Ecosystem:

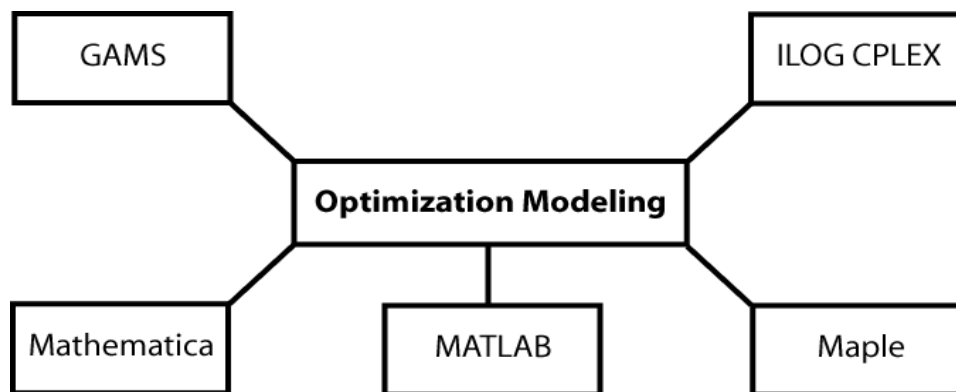


Figure 1: Overview of optimization modeling software available on the RCC systems at Penn State

1. General Algebraic Modeling System (GAMS)

GAMS is a high-level modeling system for mathematical programming and optimization. It consists of a *language compiler* and a stable of integrated high-performance *solvers*. GAMS is tailored for complex, large scale modeling applications, and allows you to build large maintainable models that can be adapted quickly to new situations. GAMS is specifically designed for modeling *linear*, *nonlinear* and *mixed integer* optimization problems. GAMS allows the user to concentrate on the modeling problem by making the setup simple. The system models problems in a highly compact and natural way. The user can change the formulation quickly and easily and can change from one solver to another seamlessly. Using GAMS, data are entered only once in familiar list and table form. Models

are described in concise algebraic statements which are natural extensions of their mathematical representation. Whole sets of closely related constraints are entered in one statement. GAMS automatically generates each constraint equation, and lets the user make exceptions in cases where generality is not desired. Statements in models can be reused without having to change the algebra when other instances of the same or related problems arise. The location and type of errors are pinpointed before a solution is attempted. Models are fully portable from one computer platform to another when GAMS is available on each platform. **Key Advantages:** Easy to use; good for rapid prototyping of one-of-a-kind problems which may require many revisions to establish an accurate model.

2. ILOG CPLEX

CPLEX algorithms can be accessed through a number of different interfaces. *CPLEX Component Libraries* include CPLEX Callable Library and ILOG Concert Technology. C, C++, C#, Java, Visual Basic and FORTRAN developers can embed powerful CPLEX algorithms within their application programs. *CPLEX Interactive Optimizer* is a command-line interactive program, provided in executable, ready-to-use form. It packs all the power and speed of CPLEX into an easy-to-use, easy-to-learn format, featuring a simple user interface and an extensive help system. ILOG's Optimization Programming Language (*ILOG OPL*) provides a natural representation of optimization models, requiring far less effort than general-purpose programming languages. ILOG OPL has advanced data types designed for the special needs of optimization problems, and it fully supports linear and quadratic objectives and constraints, as well as real and integer decision variables. It also supports both mathematical and constraint programming techniques. The ILOG OPL integrated development environment (IDE) makes it easy to evaluate different modeling approaches to a problem and to integrate external data. Debugging and tuning tools support the development process, and once ready, the model can be deployed into an external application. **Key Advantages:** Best in class solver technology for linear and integer programming models; integrating optimization models with other applications written in C, C++, Java, etc., deployment of optimization models in a production environment.

3. MATLAB (Optimization Toolbox)

The Optimization Toolbox extends the MATLAB technical computing environment with tools and widely used algorithms for standard and large-scale optimization. These algorithms solve constrained and unconstrained continuous and discrete problems. The toolbox includes functions for linear programming, quadratic programming, nonlinear optimization, nonlinear least squares, nonlinear equations, multi-objective optimization, and binary integer programming. MATLAB and Optimization Toolbox let

you easily define models, gather data, manage model formulations, and analyze results. Toolbox functions, accessible using the MATLAB command line or through a GUI, are written in the open MATLAB language. This means that you can inspect the algorithms, modify the source code, and create your own custom functions. **Key Advantages:** Familiar MATLAB programming environment; access to all other MATLAB functionalities.

4. Mathematica

Integrated into *Mathematica* are a full range of state-of-the-art local and global optimization techniques, both numeric and symbolic, including constrained nonlinear optimization, interior point methods and integer programming—as well as original symbolic methods. *Mathematica's* symbolic architecture provides seamless access to industrial-strength system and model optimization, efficiently handling million-variable linear programming, and multi-thousand-variable nonlinear problems. **Key Advantages:** Familiar Mathematica programming environment; access to all other Mathematica functionalities, ability to perform symbolic optimization.

5. Maple (Global Optimization Toolbox)

The Global Optimization Toolbox in Maple incorporates the following solver modules for nonlinear optimization problems: branch-and-bound global search, global adaptive random search, multi-start based global random search, global solution further refined by local search using the reduced gradient method. Models with thousands of variables and constraints can be solved effectively using these algorithms. The various solvers take advantage of Maple arbitrary precision capabilities in their calculations, to greatly reduce numerical instability problems. The toolbox supports arbitrary objective and constraint functions, including those defined in terms of special functions (for example, Bessel, hypergeometric), derivatives and integrals, and piecewise functions. Functions can also be defined in terms of a Maple procedure rather than a formula. Built-in model visualization capabilities exist for viewing one or two-dimensional subspace projections of the objective function, with visualization of the constraints as planes or lines on the objective surface. **Key Advantages:** Familiar Maple programming environment; access to all other Maple functionalities, ability to perform symbolic optimization.

Linear Programming Example: The Blending Problem

Several forms of gasoline are produced during the petroleum refining process, and a last step combines them to obtain market products with specified quality measures. Suppose the following four gasoline products are available:

Type	Quality Index – 1	Quality Index – 2	Cost(\$/Barrel)
1	99	210	48
2	70	335	43
3	78	280	58
4	91	265	46

Determine the minimum cost blend which has quality index-1 between 85 and 90 and quality index-2 between 270 and 280.

Mathematical Model:

minimize $48 \cdot x_1 + 43 \cdot x_2 + 58 \cdot x_3 + 46 \cdot x_4$ (Cost of Blend)

subject to

$85 \leq 99 \cdot x_1 + 70 \cdot x_2 + 78 \cdot x_3 + 91 \cdot x_4 \leq 90$ (Quality Index - 1 Specification)

$270 \leq 210 \cdot x_1 + 335 \cdot x_2 + 280 \cdot x_3 + 265 \cdot x_4 \leq 280$ (Quality Index - 2 Specification)

$x_1 + x_2 + x_3 + x_4 = 1$

$0 \leq x_1 \leq 1$ $0 \leq x_2 \leq 1$ $0 \leq x_3 \leq 1$ $0 \leq x_4 \leq 1$

(Solution : $x_1^* = 0.1765, x_2^* = 0.3529, x_3^* = 0.0000, x_4^* = 0.4706, Cost = 45.2941$)

GAMS Solution-1 (blend.gms):

```
option
lp = CoinCbc
nlp = conopt
mip = cplex
rmip = cplex
minlp = dicopt;

variables
tot_cost, x1, x2, x3, x4;

equations
objective
index_1_min, index_1_max
index_2_min, index_2_max
```

```

tot_frac;

*****Model Definition*****
objective ..
tot_cost =e= 48*x1 + 43*x2 + 58*x3 + 46*x4;

index_1_min ..
99*x1 + 70*x2 + 78*x3 + 91*x4 =g= 85;

index_1_max ..
99*x1 + 70*x2 + 78*x3 + 91*x4 =l= 90;

index_2_min ..
210*x1 + 335*x2 + 280*x3 + 265*x4 =g= 270;

index_2_max ..
210*x1 + 335*x2 + 280*x3 + 265*x4 =l= 280;

tot_frac ..
x1 + x2 + x3 + x4 =e= 1;

x1.lo = 0; x1.up = 1;
x2.lo = 0; x2.up = 1;
x3.lo = 0; x3.up = 1;
x4.lo = 0; x4.up = 1;

*****
model blend /
objective,index_1_min,index_1_max,index_2_min,index_2_max,to
t_frac/;

*****Solve Statement*****
solve blend using lp minimizing tot_cost;
*****

display tot_cost.l;
display x1.l,x2.l,x3.l,x4.l;

```

GAMS Solution-2 (blend2.gms):

```

option
lp = CoinCbc
nlp = conopt
mip = cplex
rmip = cplex
minlp = dicopt
;

sets
i      set of gasolines      /1*4/
j      set of quality indices /1*2/
;

```

```

parameters
cost(i)
quality_val(i,j)
quality_spec_min(j)
quality_spec_max(j)
;

cost('1') = 48;
cost('2') = 43;
cost('3') = 58;
cost('4') = 46;

quality_val('1','1') = 99;
quality_val('2','1') = 70;
quality_val('3','1') = 78;
quality_val('4','1') = 91;

quality_val('1','2') = 210;
quality_val('2','2') = 335;
quality_val('3','2') = 280;
quality_val('4','2') = 265;

quality_spec_min('1') = 85;
quality_spec_max('1') = 90;
quality_spec_min('2') = 270;
quality_spec_max('2') = 280;

variables
tot_cost,x(i);

equations
objective
min_spec_cons(j),max_spec_cons(j)
tot_frac_cons
;

*****Model Definition*****
objective ..
tot_cost =e= sum(i,cost(i)*x(i));

min_spec_cons(j) ..
sum(i,quality_val(i,j)*x(i)) =g= quality_spec_min(j);

max_spec_cons(j) ..
sum(i,quality_val(i,j)*x(i)) =l= quality_spec_max(j);

tot_frac_cons ..
sum(i,x(i)) =e= 1;

x.lo(i) = 0; x.up(i) = 1;
*****

```

```

model blend /
objective,min_spec_cons,max_spec_cons,tot_frac_cons/;

*****Solve Statement*****
solve blend using lp minimizing tot_cost;
*****

file res /blend_2.res/;
put res;

put "Total Cost = ",tot_cost.l:8:4/;
loop(i,
put "x(",i.tl:1:0,") = ",x.l(i):8:4/;
);

```

ILOG CPLEX Interactive Optimizer Solution:

```

\Problem name:blend
Minimize
  obj: 48 x1 + 43 x2 + 58 x3 + 46 x4
Subject To
  c1: 99 x1 + 70 x2 + 78 x3 + 91 x4 >= 85
  c2: 99 x1 + 70 x2 + 78 x3 + 91 x4 <= 90
  c3: 210 x1 + 335 x2 + 280 x3 + 265 x4 >= 270
  c4: 210 x1 + 335 x2 + 280 x3 + 265 x4 <= 280
  c5: x1 + x2 + x3 + x4 = 1
Bounds
  0 <= x1 <= 1
  0 <= x2 <= 1
  0 <= x3 <= 1
  0 <= x4 <= 1
End

```

MATLAB Solution:

```

f = [48;43;58;46];

A = [ 99 70 78 91
      -99 -70 -78 -91
        210 335 280 265
      -210 -335 -280 -265];
b = [90; -85; 280; -270];

lb = [0; 0; 0; 0];
ub = [1; 1; 1; 1];

Aeq = [1 1 1 1];
beq = [1];

options=optimset('Display','iter');
[x,objective,exitflag,output,lambda]=
linprog(f,A,b,Aeq,beq,lb,ub,[],options)

```

Mathematica Solution-1 (Algebraic Form):

```
NMinimize[{48*x1 + 43*x2 + 58*x3 + 46*x4,  
85 <= 99*x1 + 70*x2 + 78*x3 + 91*x4 <= 90 &&  
270 <= 210*x1 + 335*x2 + 280*x3 + 265*x4 <= 280 &&  
x1 + x2 + x3 + x4 == 1 &&  
0 <= x1 <= 1 && 0 <= x2 <= 1 && 0 <= x3 <= 1 && 0 <= x4 <= 1},  
{x1, x2, x3, x4}]
```

Mathematica Solution-2 (Matrix Form):

```
c={48, 43, 58, 46};  
m={{99, 70, 78, 91}, {99, 70, 78, 91}, {210, 335, 280, 265},  
{210, 335, 280, 265}, {1, 1, 1, 1}};  
b={{90, -1}, {85, 1}, {280, -1}, {270, 1}, {1, 0}};  
N[LinearProgramming[c, m, b, {{0, 1}, {0, 1}, {0, 1}, {0, 1}}]]
```

Maple Solution:

```
with(Optimization):  
LPSolve(48*x1 + 43*x2 + 58*x3 + 46*x4,  
{85 <= 99*x1 + 70*x2 + 78*x3 + 91*x4,  
99*x1 + 70*x2 + 78*x3 + 91*x4 <= 90,  
270 <= 210*x1 + 335*x2 + 280*x3 + 265*x4,  
210*x1 + 335*x2 + 280*x3 + 265*x4 <= 280,  
x1 + x2 + x3 + x4 = 1},  
x1=0..1, x2=0..1, x3=0..1, x4=0..1);
```

Integer Programming Example: The Eight Queens Puzzle

The eight queens puzzle is the problem of putting eight chess queens on an 8×8 chessboard such that none of them is able to capture any other using the standard chess queen's moves. The color of the queens is meaningless in this puzzle, and any queen is assumed to be able to attack any other. *Thus, a solution requires that no two queens share the same row, column, or diagonal.* The eight queens puzzle is an example of the more general n queens puzzle of placing n queens on an n×n chessboard.

Sets:

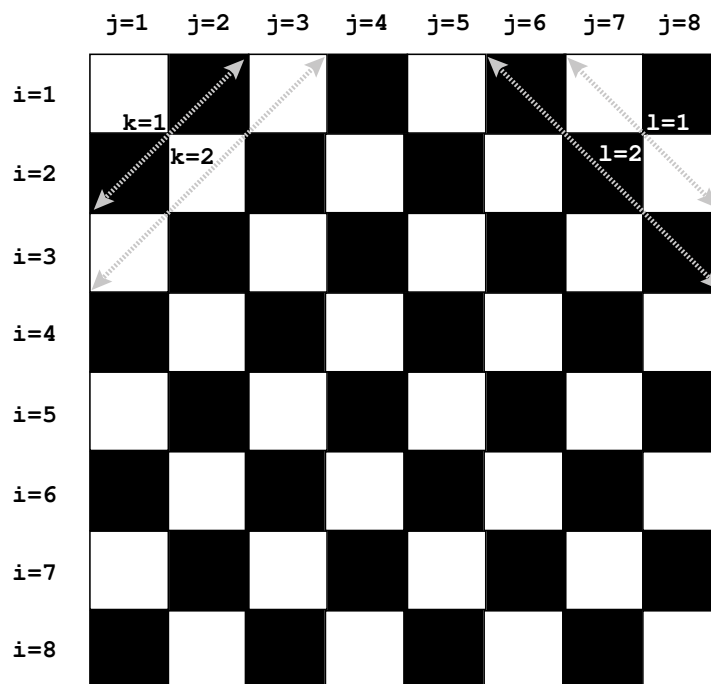
$i = \{1, 2, \dots, 8\}$ = set of rows

$j = \{1, 2, \dots, 8\}$ = set of columns

$k = \{1, 2, \dots, 13\}$ = set of diagonals - 1

$l = \{1, 2, \dots, 13\}$ = set of diagonals - 2

Parameters:



$a_{ijk} = 1$ if position (i, j) belongs to diagonal k
 $= 0$ otherwise

$b_{ijl} = 1$ if position (i, j) belongs to diagonal l
 $= 0$ otherwise

$a_{i=2, j=1, k=1} = 1, a_{i=1, j=2, k=1} = 1, a_{i=3, j=1, k=2} = 1, a_{i=2, j=2, k=2} = 1, a_{i=1, j=3, k=2} = 1, \dots$

$b_{i=8, j=2, l=1} = 1, b_{i=7, j=1, l=1} = 1, b_{i=8, j=3, l=2} = 1, b_{i=7, j=2, l=2} = 1, b_{i=6, j=1, l=2} = 1, \dots$

Discrete Decision Variables:

$Y_{ij} = 1$ if a queen is placed at position (i, j)
 $= 0$ otherwise

Model Formulation:

$$\text{Maximize}_{Y_{ij} \in \{0,1\}} \sum_{i=1}^8 \sum_{j=1}^8 Y_{ij} \quad \text{(Total number of queens)}$$

subject to

$$\sum_{j=1}^8 Y_{ij} \leq 1 \quad \forall i \quad \text{(At most one queen in each row)}$$

$$\sum_{i=1}^8 Y_{ij} \leq 1 \quad \forall j \quad \text{(At most one queen in each column)}$$

$$\sum_{i=1}^8 \sum_{j=1}^8 a_{ijk} \cdot Y_{ij} \leq 1 \quad \forall k \quad \text{(At most one queen in each diagonal-1)}$$

$$\sum_{i=1}^8 \sum_{j=1}^8 b_{ijl} \cdot Y_{ij} \leq 1 \quad \forall l \quad \text{(At most one queen in each diagonal-2)}$$

GAMS Solution:

```
option
lp = cplex
nlp = conopt
mip = CoinCbc
rmip = cplex
minlp = dicopt
;
```

sets

```
i      set of rows      /1*8/
j      set of columns   /1*8/
```

```

k          set of diagonal-1          /1*13/
l          set of diagonal-2          /1*13/;

parameters
a(i,j,k)
b(i,j,l);

$include data.txt

variables
z;

binary variables
y(i,j);

equations
objective
row_const(i)
col_const(j)
diag1_const(k)
diag2_const(l);

*****Model Equations*****
objective ..
row_const(i) ..
col_const(j) ..
diag1_const(k) ..
diag2_const(l) ..

*****

*****Model Definition*****
objective ..
z =e= sum((i,j), y(i,j));

row_const(i) ..
sum(j,y(i,j)) =l= 1;

col_const(j) ..
sum(i,y(i,j)) =l= 1;

diag1_const(k) ..
*sum((i,j), a(i,j,k)*y(i,j)) =l= 1;
sum((i,j)$(a(i,j,k) eq 1), y(i,j)) =l= 1;

diag2_const(l) ..
sum((i,j), b(i,j,l)*y(i,j)) =l= 1;
*****

*****Model Solve*****
model queens / objective,row_const,col_const,diag1_const,diag2_const/;
solve queens using mip maximizing z;
*****

file res /queens.res/;
put res;
put "---- # Queens = ",z.l:1:0" ----"/;
loop(i,
loop(j,
put y.l(i,j):2:0;
);
put /;
);

```

A Solution:

	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8
i=1							Q	
i=2					Q			
i=3			Q					
i=4	Q							
i=5						Q		
i=6								Q
i=7		Q						
i=8				Q				

- Total of 92 possible solutions
- 12 unique solutions (rotation and reflection operations generate rest)