Using the OPTMODEL Procedure in SAS/OR to Solve Complex Problems

Rob Pratt, Senior R&D Manager, SAS/OR
Outline

1. Recent Features in PROC OPTMODEL
2. Graph Partitioning with Connectivity Constraints
SUBMIT Block

- Access the full SAS system without exiting OPTMODEL
- Build novel dynamic algorithms using statistics, matrix manipulation, visualization, interaction, etc.

Syntax:

```
proc optmodel;
...
submit [arguments] [/ options];
[arbitrary SAS code]
endsubmit;
...
quit;
```
Network Algorithms

- PROC OPTNET: specialized procedure
- PROC OPTMODEL access using:
  - SUBMIT block
  - SOLVE WITH NETWORK (new in SAS/OR 13.1)

- Connected components
- Biconnected components and articulation points
- Maximal cliques
- Cycles
- Transitive closure
- Linear assignment problem

- Shortest path problem
- Minimum-cost network flow problem
- Minimum spanning tree problem
- Minimum cut problem
- Traveling salesman problem
**Constraint Programming Solver**

- PROC CLP: specialized procedure

- PROC OPTMODEL access using:
  - SUBMIT block
  - SOLVE WITH CLP (new in SAS/OR 13.2)

- Supported features and predicates
  - Multiple solutions
  - Strict inequalities ($<$, $>$) and disequality ($\neq$)
  - ALLDIFF
  - ELEMENT
  - GCC
  - LEXICO
  - PACK
  - REIFY
COFOR Loop

- Uses multiple threads to execute SOLVE statements in parallel across iterations of loop

- Syntax for serial FOR loop:
  ```
  for {i in ISET} do;
      ...
      solve ...;
      ...
  end;
  ```

- Syntax for parallel COFOR loop:
  ```
  cofor {i in ISET} do;
      ...
      solve ...;
      ...
  end;
  ```
Problem Description

- Given undirected graph, node weights, positive integer $k$, target $t \in [\ell, u]$
- Partition node set into $k$ subsets, each with total weight within $[\ell, u]$.
- Usual objective: minimize sum of edge weights between parts of partition.
- Our objective: find most “balanced” partition
  - Minimize total absolute deviation of subset weight from target ($L_1$ norm).
  - Minimize maximum absolute deviation of subset weight from target ($L_\infty$ norm).
- Connectivity constraints: each of the $k$ resulting induced subgraphs must be connected.
Motivation

  - Political districting
- Advanced Analytics and Optimization Services consulting engagement
  - Assigning service workers to customer regions
- INFORMS OR Exchange posting (Jan. 2013)
  - Forestry

Map from Shasha (2002)
Map Represented as Graph
Node Weights (Populations)
Feasible Solution for 14 Clusters, Bounds $12000 \pm 100$
Solution Approaches

- Compact MILP formulation with $IsCapital[i]$ binary variables and nonnegative flow variables
- Row generation, dynamically excluding disconnected subsets
- Column generation, dynamically generating connected subsets
- Column generation, statically generating all possible connected subsets
Solution Approaches

- Compact MILP formulation with $IsCapital[i]$ binary variables and nonnegative flow variables
- Row generation, dynamically excluding disconnected subsets
- Column generation, dynamically generating connected subsets
- Column generation, statically generating all possible connected subsets
Master MILP problem: Given all feasible subsets, solve cardinality constrained set partitioning problem

/* master problem: partition nodes into connected subsets, */
/* minimizing deviation from target */
/* UseSubset[s] = 1 if subset s is used, 0 otherwise */
var UseSubset {SUBSETS} binary;

/* L_1 norm */
num subset_population {s in SUBSETS} = sum {i in NODES_s[s]} p[i];
num subset_deviation {s in SUBSETS} = abs(subset_population[s] - target);
min Objective1 = sum {s in SUBSETS} subset_deviation[s] * UseSubset[s];

/* each node belongs to exactly one subset */
con Partition {i in NODES}:
  sum {s in SUBSETS: i in NODES_s[s]} UseSubset[s] = 1;

/* use exactly the allowed number of subsets */
con Cardinality:
  sum {s in SUBSETS} UseSubset[s] = num_clusters;

problem Master include
  UseSubset Objective1 Partition Cardinality;
Static Column Generation

1. Generate all possible connected subsets with weights close enough to target
   - Relax connectivity and find all possible subsets
   - Postprocess to filter out disconnected subsets
2. Solve cardinality constrained set partitioning master problem
Static Column Generation

1. Generate all possible connected subsets with weights close enough to target
   1. Relax connectivity and find all possible subsets
   2. Postprocess to filter out disconnected subsets

2. Solve cardinality constrained set partitioning master problem
Generate all possible connected subsets with weights close enough to target

1. Relax connectivity and find all possible subsets
2. Postprocess to filter out disconnected subsets

2. Solve cardinality constrained set partitioning master problem
Features Used in PROC OPTMODEL

- MILP solver to compute maximum subset size for each node (COFOR loop)
- Network solver (all-pairs shortest path) to generate valid cuts that encourage connectivity
- Constraint programming solver to generate subsets (multiple solutions)
- Network solver (connected components) to check connectivity (COFOR loop)
- MILP solver to solve set partitioning problem
Given node populations $p_i$ and target population range $[\ell, u]$

- Find all connected node subsets close to target population
- Binary variable $x_i$ indicates whether node $i$ appears in subset

$$\ell \leq \sum_{i \in N} p_i x_i \leq u$$

- Connectivity hard to impose directly, so relax (but do not ignore)
Valid Cuts for Nodes Too Far Apart

- $m_i$: max # nodes among feasible subsets that contain node $i$
- If shortest path distance $d_{ij} \geq \min(m_i, m_j)$ then $x_i + x_j \leq 1$
/* maxnodes[i] = max number of nodes among feasible subsets that contain node i */
num maxnodes {NODES};

/* UseNode[i] = 1 if node i is used, 0 otherwise */
var UseNode {NODES} binary;

max NumUsed = sum {i in NODES} UseNode[i];

con PopulationBounds:
  target - deviation_ub
  <= sum {i in NODES} p[i] * UseNode[i]
  <= target + deviation_ub;

problem MaxNumUsed include
  UseNode NumUsed PopulationBounds;
use problem MaxNumUsed;

put 'Finding max number of nodes among feasible subsets that contain node i...';
cofor {i in NODES} do;
  put i=;
  fix UseNode[i] = 1;
  solve with MILP / loglevel=0 logfreq=0;
  maxnodes[i] = round(NumUsed.sol);
  unfix UseNode;
end;
/* distance[i,j] = shortest path distance (number of edges) from i to j */
num distance {NODES, NODES};
/* num one {EDGES} = 1; */
put 'Finding all-pairs shortest path distances...';
solve with NETWORK /
    links       = (include=EDGES /*weight=one*/)
    shortpath   = (spweights=distance);
Valid Cuts for Small-Population Nodes

- If $x_i = 1$ and $p_i < \ell$ then $x_j = 1$ for some neighbor $j$ of $i$
- $x_i \leq \sum_{j \in N_i} x_j$
Valid Cuts for Pairs of Small-Population Nodes

- If $x_i = x_j = 1$ and $p_i + p_j < \ell$ then $x_k = 1$ for some neighbor $k$ in moat around \{i, j\}
- $x_i + x_j - 1 \leq \sum_{k \in (N_i \cup N_j) \setminus \{i, j\}} x_k$
Valid Cuts for Pairs of Small-Population Nodes

- Can replace this cut with two stronger cuts
- If \( x_i = 1 \) or \( x_j = 1 \) and \( p_i + p_j < \ell \) then \( x_k = 1 \) for some neighbor \( k \) in moat around \( \{i, j\} \)
  \[
  x_i \leq \sum_{k \in (N_i \cup N_j) \setminus \{i, j\}} x_k
  \]
  \[
  x_j \leq \sum_{k \in (N_i \cup N_j) \setminus \{i, j\}} x_k
  \]
Valid Cuts for Sets with Small Population

- More generally, if $x_i = 1$ for some $i \in S \subset N$ such that $\sum_{j \in S} p_j < \ell$
  
  then $x_k = 1$ for some neighbor $k$ in moat around $S$

- $x_i \leq \sum_{k \in (\cup_{j \in S} N_j) \setminus S} x_k$

- Known as “ring inequalities” in the literature
set NODE_PAIRS = {i in NODES, j in NODES: i < j};

/* cannot have two nodes i and j that are too far apart */
con Conflict {<i,j> in NODE_PAIRS: distance[i,j] >= min(maxnodes[i],maxnodes[j])}:
    UseNode[i] + UseNode[j] <= 1;

/* if UseNode[i] = 1 and p[i] < target - deviation_ub 
    then UseNode[j] = 1 for some neighbor j of i */
con Moat1 {i in NODES: p[i] < target - deviation_ub}:
    UseNode[i] <= sum {j in NEIGHBORS[i]} UseNode[j];

/* if (UseNode[i] = 1 or UseNode[j] = 1) and p[i] + p[j] < target - deviation_ub 
    then UseNode[k] = 1 for some neighbor k in moat around (i,j) */
con Moat2_i {<i,j> in NODE_PAIRS: p[i] + p[j] < target - deviation_ub}:
    UseNode[i] <= sum {k in (NEIGHBORS[i] union NEIGHBORS[j]) diff {i,j}} UseNode[k];
con Moat2_j {<i,j> in NODE_PAIRS: p[i] + p[j] < target - deviation_ub}:
    UseNode[j] <= sum {k in (NEIGHBORS[i] union NEIGHBORS[j]) diff {i,j}} UseNode[k];

problem Subproblem include
    UseNode PopulationBounds Conflict Moat1 Moat2_i Moat2_j;
use problem Subproblem;

put 'Solving CLP subproblem...';
solve with CLP / findallsolns;
/* use _NSOL_ and .sol[s] to access multiple solutions */
SUBSETS = 1.._NSOL_;
for {s in SUBSETS} NODES_s[s] = {i in NODES: UseNode[i].sol[s] > 0.5};
NOTE: Problem generation will use 4 threads.
NOTE: The problem has 66 variables (0 free, 0 fixed).
NOTE: The problem has 66 binary and 0 integer variables.
NOTE: The problem has 5123 linear constraints (5122 LE, 0 EQ, 0 GE, 1 range).
NOTE: The problem has 42926 linear constraint coefficients.
NOTE: The problem has 0 nonlinear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The experimental CLP solver is called.
NOTE: All possible solutions have been found.
NOTE: Number of solutions found = 4472.

CARD(SUBSETS) = 4472
/* check connectivity */
set NODES_THIS;
num component {NODES_THIS};
set DISCONNECTED init {};
cofor {s in SUBSETS} do;
   put s =;
   NODES_THIS = NODES_s[s];
   if card(NODES_THIS) <= 3 then continue; /* connected */
   solve with NETWORK /
       links = (include=EDGES)
       subgraph = (nodes=NODES_THIS)
       concomp
       out = (concomp=component);
   if or {i in NODES_THIS} (component[i] > 1)
      then DISCONNECTED = DISCONNECTED union {s};
   end;
SUBSETS = SUBSETS diff DISCONNECTED;
Network Solver Log

s=1
NOTE: The SUBGRAPH= option filtered 145 elements from 'EDGES.'
NOTE: The number of nodes in the input graph is 4.
NOTE: The number of links in the input graph is 3.
NOTE: Processing connected components.
NOTE: The graph has 1 connected component.
NOTE: Processing connected components used 0.00 (cpu: 0.00) seconds.
...
s=28
NOTE: The SUBGRAPH= option filtered 144 elements from 'EDGES.'
NOTE: The number of nodes in the input graph is 6.
NOTE: The number of links in the input graph is 4.
NOTE: Processing connected components.
NOTE: The graph has 2 connected components.
NOTE: Processing connected components used 0.00 (cpu: 0.00) seconds.
...
s=4472
NOTE: The SUBGRAPH= option filtered 144 elements from 'EDGES.'
NOTE: The number of nodes in the input graph is 5.
NOTE: The number of links in the input graph is 4.
NOTE: Processing connected components.
NOTE: The graph has 1 connected component.
NOTE: Processing connected components used 0.00 (cpu: 0.00) seconds.

CARD(SUBSETS)=2771
/* master problem: partition nodes into connected subsets, */
/* UseSubset[s] = 1 if subset s is used, 0 otherwise */
var UseSubset {SUBSETS} binary;

/* L_1 norm */
num subset_population {s in SUBSETS} = sum {i in NODES_s[s]} p[i];
num subset_deviation {s in SUBSETS} = abs(subset_population[s] - target);
min Objective = sum {s in SUBSETS} subset_deviation[s] * UseSubset[s];

/* each node belongs to exactly one subset */
con Partition {i in NODES}:
    sum {s in SUBSETS: i in NODES_s[s]} UseSubset[s] = 1;

/* use exactly the allowed number of subsets */
con Cardinality:
    sum {s in SUBSETS} UseSubset[s] = num_clusters;

problem Master include
    UseSubset Objective Partition Cardinality;

...
NOTE: Problem generation will use 4 threads.
NOTE: The problem has 2771 variables (0 free, 0 fixed).
NOTE: The problem has 2771 binary and 0 integer variables.
NOTE: The problem has 67 linear constraints (0 LE, 67 EQ, 0 GE, 0 range).
NOTE: The problem has 17334 linear constraint coefficients.
NOTE: The problem has 0 nonlinear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The OPTMODEL presolver is disabled for linear problems.
NOTE: The MILP presolver value AUTOMATIC is applied.
NOTE: The MILP presolver removed 1092 variables and 0 constraints.
NOTE: The MILP presolver removed 7089 constraint coefficients.
NOTE: The MILP presolver modified 0 constraint coefficients.
NOTE: The presolved problem has 1679 variables, 67 constraints, and 10245 constraint coefficients.
NOTE: The MILP solver is called.

<table>
<thead>
<tr>
<th>Node</th>
<th>Active</th>
<th>Sols</th>
<th>BestInteger</th>
<th>BestBound</th>
<th>Gap</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>.</td>
<td>96.3214286</td>
<td>.</td>
<td>1</td>
</tr>
</tbody>
</table>

... 

<table>
<thead>
<tr>
<th>Node</th>
<th>Active</th>
<th>Sols</th>
<th>BestInteger</th>
<th>BestBound</th>
<th>Gap</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>.</td>
<td>113.1320755</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>124.0000000</td>
<td>117.1666667</td>
<td>5.83%</td>
<td>2</td>
</tr>
</tbody>
</table>

NOTE: The MILP solver added 40 cuts with 2752 cut coefficients at the root.

<table>
<thead>
<tr>
<th>Node</th>
<th>Active</th>
<th>Sols</th>
<th>BestInteger</th>
<th>BestBound</th>
<th>Gap</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>124.0000000</td>
<td>124.0000000</td>
<td>0.00%</td>
<td>2</td>
</tr>
</tbody>
</table>

NOTE: Optimal.
NOTE: Objective = 124.
Solution for 14 Clusters, Bounds $12000 \pm 100$, Objective 1
/* L_infinity norm */
var MaxDeviation >= 0;
min Objective2 = MaxDeviation;
con MaxDeviationDef {s in SUBSETS}:
    MaxDeviation >= subset_deviation[s] * UseSubset[s];

put 'Solving master problem, minimizing Objective2...';
MaxDeviation = max {s in SUBSETS} (subset_deviation[s] * UseSubset[s].sol);
solve with MILP / primalin;
MILP Solver Log

NOTE: The problem has 2772 variables (0 free, 0 fixed).
NOTE: The problem has 2771 binary and 0 integer variables.
NOTE: The problem has 2838 linear constraints (0 LE, 67 EQ, 2771 GE, 0 range).
NOTE: The problem has 22845 linear constraint coefficients.
NOTE: The problem has 0 nonlinear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The MILP presolver value AUTOMATIC is applied.
NOTE: The MILP presolver removed 1092 variables and 1110 constraints.
NOTE: The MILP presolver removed 9278 constraint coefficients.
NOTE: The MILP presolver modified 0 constraint coefficients.
NOTE: The presolved problem has 1680 variables, 1728 constraints, and 13567 constraint coefficients.
NOTE: The MILP solver is called.

<table>
<thead>
<tr>
<th>Node</th>
<th>Active</th>
<th>Sols</th>
<th>BestInteger</th>
<th>BestBound</th>
<th>Gap</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>37.00000000</td>
<td>5.8468795</td>
<td>532.82%</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>37.00000000</td>
<td>26.0503748</td>
<td>42.03%</td>
<td>4</td>
</tr>
</tbody>
</table>

NOTE: The MILP solver added 89 cuts with 4657 cut coefficients at the root.

| 3    | 1      | 2    | 36.00000000 | 29.00000000 | 24.14% | 5    |
| 6    | 2      | 3    | 35.00000000 | 29.00000000 | 20.69% | 5    |
| 7    | 1      | 5    | 35.00000000 | 29.00000000 | 20.69% | 6    |
| 64   | 54     | 6    | 29.00000000 | 29.00000000 | 0.00%  | 7    |

NOTE: Optimal.
NOTE: Objective = 29.
Solution for 14 Clusters, Bounds $12000 \pm 100$, Objective 2
Live Demo

OPTMODEL code
GraphPartitioningConnectivity.sas
Using the OPTMODEL Procedure in SAS/OR to Solve Complex Problems