Paper 203-2008

# Using SAS/OR® and SAS® Grid Manager to Solve Optimization Problems on the Grid

Matthew Galati, SAS Institute Inc., Wayne, PA
Doug Haigh, Rob Pratt, Ivan Oliveira, SAS Institute Inc., Cary, NC

## ABSTRACT

Some of the most demanding optimization problems occurring today can test the limits of even the fastest optimization solvers. For such problems, finding an optimal solution or even a good solution quickly can be quite difficult. Grid computing, which enables the distribution of challenging computational tasks across multiple CPUs, is one answer. SAS/OR, featuring new optimization solvers and the OPTMODEL procedure, can work together with SAS Grid Manager to provide grid-enabled optimization.

This paper will provide a few examples of how SAS/OR code can be moved easily to a grid environment through the new grid-enabled version of the %Distribute macro that makes use of SAS Grid Manager. SAS Grid Manager helps automate the management of SAS computing grids with dynamic load balancing, resource assignment, job priority, and termination management.

We will provide an introduction to this technology by exploring an application in the area of facility location analysis. We will discuss the serial implementation that uses just SAS/OR, and we will demonstrate the advantages of using grid-enabled versions of the code.

## INTRODUCTION

The field of *Operations Research* (OR) is a multi-disciplinary branch of mathematics with applications in almost every major industry. One of the most common tools used by OR practitioners is mathematical programming, or optimization. *Optimization* is the science of attempting to find the minimum (or maximum) of some function where the domain is constrained to some set of candidate solutions. Often this domain is defined compactly as a set of inequalities which constrains the solution space. SAS/OR provides a suite of procedures that enable you to solve various types of optimization problems. In general, finding the optimal solution to an optimization problem with a domain that is non-convex is very difficult. These types of problems are known to be in the complexity class NP-hard (Garey and Johnson 1979), and therefore pose a significant computational challenge for any optimization software vendor.

The idea of distributed processing for analytics has been around for quite some time. SAS has been able to run its code concurrently on multiple computers for many years through its SAS/CONNECT product. In recent years, using clusters of commodity-priced machines has become much more popular. In order to meet this demand, SAS has created a new product called SAS Grid Manager which combines the power of SAS/CONNECT with the load balancing, resource assignment, job priority, and policy management capabilities of grid middleware such as Platform Computing's Load Sharing Facility (LSF). Platform LSF is included with SAS Grid Manager as Platform Suite for SAS.

The marriage of grid computing and optimization is a new and exciting area of research. Most optimization problems are too difficult to solve directly. Instead, practitioners often rely on heuristics that are based on optimization solvers as components of a bigger algorithm. Many times these heuristics lend themselves well to decomposition. Any time the algorithm or data can be decomposed in some natural way, the computation might be able to be distributed across multiple processors.

Some common examples of optimization-related tasks that can take advantage of distributed processing are:

- Monte Carlo methods (Fishman 1995)
- decomposition methods such as Dantzig-Wolfe, Benders, Lagrangian relaxation (Ralphs and Galati 2005)
- multi-start methods for global optimization (Horst and Tuy 1993)
- genetic and evolutionary algorithms (Levine 1994)
- scenario analysis
- parameter tuning
- simulation (Robinson 2004)

This paper presents the first example, Monte Carlo methods, as applied to a classical OR location problem known as the multisource Weber problem. In our experiments we combine the analytical power of SAS/OR with the grid management framework provided by SAS Grid Manager.

## LOCATION ANALYSIS

The use of location analysis by OR professionals is common in many industries. There is an abundance of different types of location models, all of which address a variant of the question of where to place entities and how to utilize them to minimize some objective, such as cost.

### WEBER PROBLEM

One of the most famous problems in location analysis is the *Weber problem*. In this problem, one is given a set, $C$, of points in Euclidean $n$-space. To give the problem some context, let us assume $n = 2$ and call these points *customers*. The goal of the Weber problem is to locate another point, call this the *facility*, which has the property of minimizing the sum of the Euclidean distances from the customers.

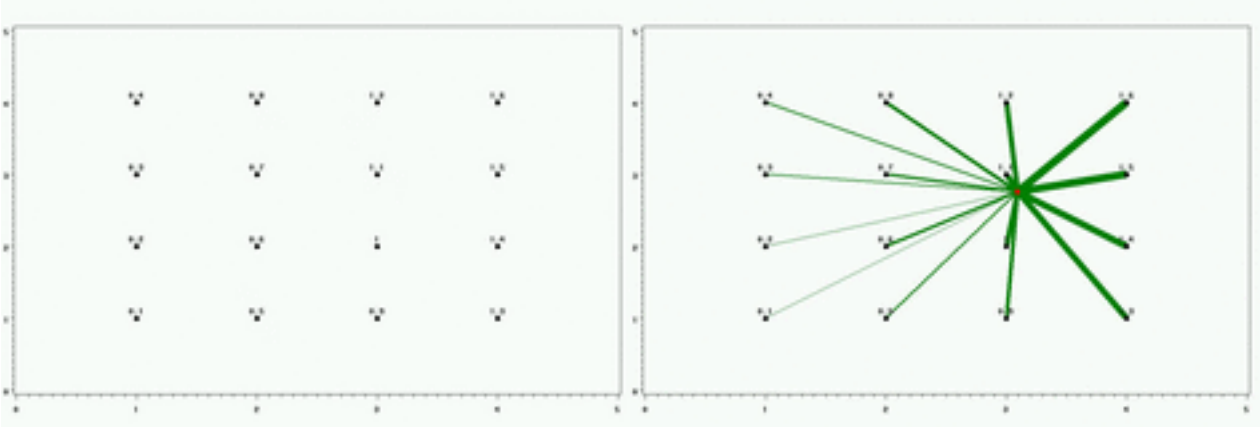
Let continuous decision variables $x = (x^1, x^2) \in \mathbf{R}^2$ define the coordinates of the facility to be located and input data $a_i = (a_i^1, a_i^2) \in \mathbf{R}^2$, for $i \in C$, define the coordinates of the customers. For each customer, let us also associate some cost $c_i$, for $i \in C$. Then, we can model the Weber problem as the following unconstrained mathematical program.

$$\min_{x \in \mathbf{R}^2} \quad \sum_{i \in C} c_i \, ||x - a_i||$$

This problem is relatively easy, because it is an unconstrained convex optimization problem. Using routines from the SAS/OR nonlinear programming packages, we can solve this problem easily.

In Figure 1 we show a small example of the Weber problem with $|C| = 16$ customers and a solution. In the solution, the red dot indicates the placement of the facility and the thickness of the green lines represents the flow of demand from each customer to the facility.

**Figure 1** Example Customer Demand and Solution for Weber problem $|C| = 16$



### MULTISOURCE WEBER PROBLEM

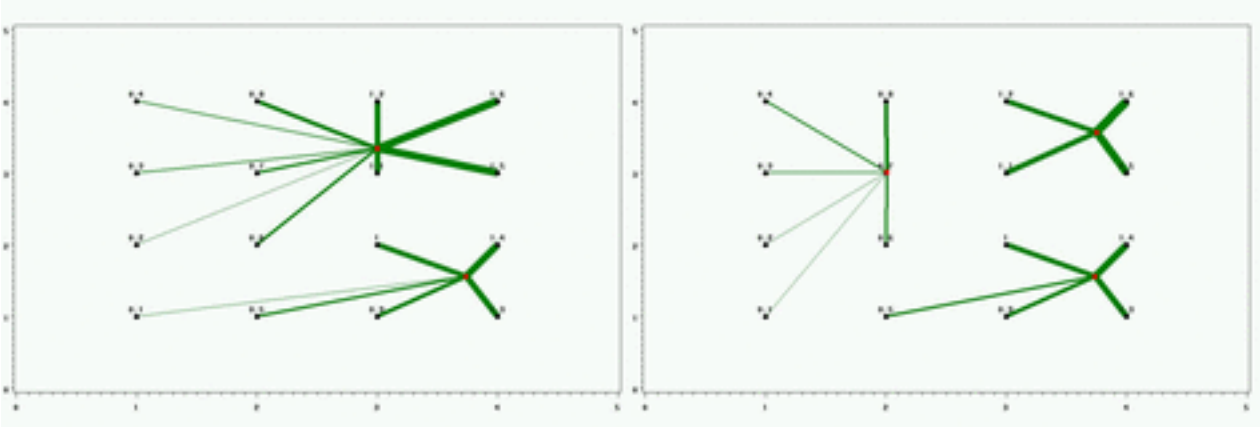
A more interesting problem in location analysis is the *multisource Weber problem* (MSWP). This is simply a generalization of the Weber problem to allow for locating a set of $m$ facilities. Define another set $F$, with $|F| = m$, as the set of facilities. Let continuous decision variables $x_j = (x_j^1, x_j^2) \in \mathbf{R}^2$, for $j \in F$, define the coordinates of the facility to be located. In addition, let us assume we are given a demand at each customer; call this $d_i$, for $i \in C$. Then, let $w_{ij} \geq 0$ define a continuous decision variable that denotes the amount of demand at customer $i \in C$ satisfied by facility $j \in F$. Now, we can model the multisource Weber problem as the following mathematical program.

$$
\begin{array}{rlcll}
\min & \sum_{i \in C, j \in F} w_{ij} \, ||x_j - a_i|| & & & \\
\text{subject to} & \sum_{j \in F} w_{ij} & = & d_i, & \forall i \in C \\
& x_j & & \text{free}, & \forall j \in F \\
& w_{ij} & \geq & 0, & \forall i \in C, j \in F
\end{array}
$$

This problem is much more difficult than the single-source version because it is no longer a convex optimization problem. In fact, this problem is well-known to have numerous poor-quality local optima where solvers can get trapped. For example, in Eilon ($|C| = 50, |F| = 5$) (S. Eilon and Christofides 1969), they showed that there were 61 local minima whose best-to-worst solutions ranged in optimality by 40.9%. This problem is extremely difficult to solve globally.

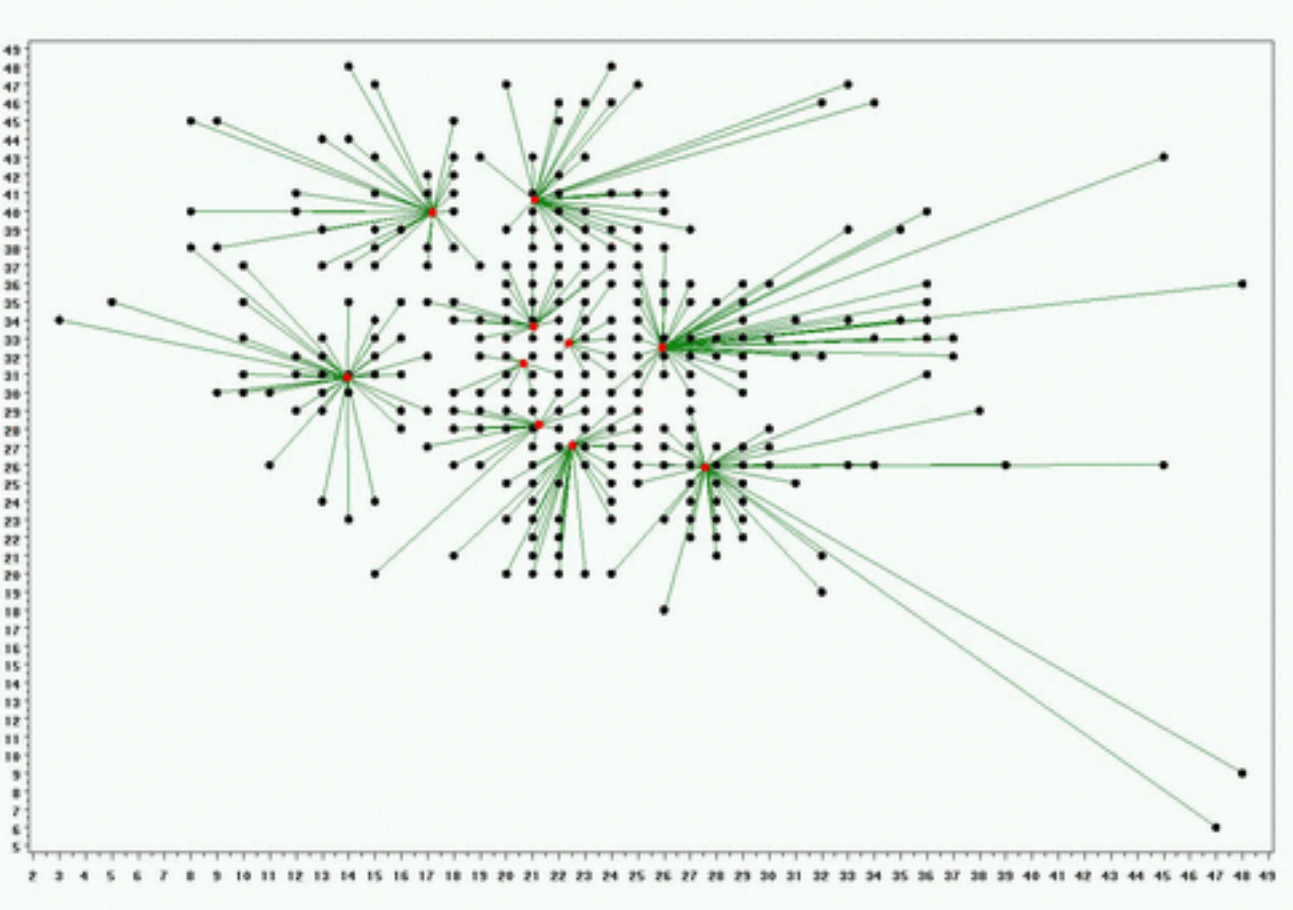
In Figure 2 we now show a small example of the multisource Weber problem with $|C| = 16$ customers and a solution for $m = 2$ and $m = 3$. Once again, the red dots indicate the placement of the facilities and the thickness of the green lines represents the flow of demand from each customer to a facility.

**Figure 2** Example Solution for Weber problem with $|C| = 16$, $m = 2$, and $m = 3$



The Section "COMPUTATIONAL RESULTS" on page 7 we introduce some real-world examples of the multisource Weber problem. A well-known example *bon287* has $|C| = 287$ customers (Taillard 2007). In Figure 3 we show a near-optimal solution to bon287 for $m = 10$. In this case, the flow is not represented by the thickness of the green lines, since this would obscure the graphic.

**Figure 3** Example Solution for Weber problem with $|C| = 287$ and $m = 10$



The multisource Weber problem has applications in numerous industry settings. Some examples that have appeared in recent literature include choosing locations for the following items (Hansen, Mladenovic̀, and Taillard 1998) :

- public, industrial, and commercial service centers
- retail outlets, schools, hospitals warehouses and plants
- trees in landscape conservation
- territories in territory design
- water sources in water source management

Due to the importance of this problem, it is worth the effort to generate heuristic approaches to solving the problem. The next section discusses the method we use which takes full advantage of the technologies in both SAS/OR and SAS Grid Manager to provide a very effective grid-enabled heuristic to this important problem.

## HEURISTIC METHODS

One heuristic approach to the multisource Weber problem attempts to discretize the location solution space $\mathbf{R}^2$. That is, by selecting a finite set of coordinates as candidate facility locations, we can reformulate the problem as a *mixed-integer linear program* (MILP).

Let $L$ denote a finite set of candidate facility locations. Now, input data $x_j = (x_j^1, x_j^2) \in \mathbf{R}^2$, for $j \in L$, define given coordinates of the candidate facilities. In addition, we now introduce a binary decision variable $y_j \in \{0, 1\}$, for $j \in L$,

which denotes whether a facility is located at candidate site $j$. Then, we can model an approximation to the multisource Weber problem as the following MILP, also known as the *$p$-median problem*.

$$\begin{array}{rrcll} \min & \sum_{i\in C, j\in L} w_{ij}\,||x_j - a_i|| & & & \\ \text{subject to} & \sum_{j\in L} w_{ij} & = & d_i, & \forall i \in C \\ & \sum_{j\in L} y_j & = & |F| & \\ & w_{ij} & \leq & d_i y_j, & \forall i \in C, j \in L \\ & y_j & \in & \{0,1\}, & \forall j \in L \\ & w_{ij} & \geq & 0, & \forall i \in C, j \in L \end{array}$$

For relatively small $|L|$, the SAS/OR MILP solver in OPTMODEL can easily solve the $p$-median problem. For larger candidate sets, this problem becomes too big. Clearly, as $|L| \to \infty$, the MILP formulation provides an increasingly better approximation of the original nonlinear program.

### *p*-MEDIAN PROBLEM

To choose $L$, one can uniformly randomly sample $\mathbf{R}^2$. For very large $L$, this should provide a good candidate set. However, choosing a large set $L$ is prohibitive because the size of the MILP will grow too big. Instead, one heuristic suggested in the literature (Hansen, Mladenovic̀, and Taillard 1998) is to choose the set of candidate facilities as the set of customer locations. That is, let $L = C$. The motivation behind the heuristic is that we want to a locate facilities close to some customer. This simple idea works surprisingly well as evidenced in Section "COMPUTATIONAL RESULTS" on page 7.

### POST PROCESSING

The solution of the $p$-median problem provides a clustering of customers to facilities, as designated by the solution vector $w^*$. Given an assignment of customer $i$ to facility $j$, we can improve the placement of $j$ by solving an auxiliary nonlinear program (NLP). Let $C_j$ represent the set of customers assigned to facility $j \in L$. Then to optimize the placement of that facility $j$, we solve the following unconstrained nonlinear program.

$$\min_{x\in\mathbf{R}^2} \quad \sum_{i\in C_j} w^*_{ij}\,||x_j - a_i||$$

This problem can be solved independently for each $j \in \{l \in L \mid y_l = 1\}$. Notice that each of these is equivalent to solving the (single-source) Weber problem discussed in Section "WEBER PROBLEM" on page 2. Using the SAS/OR NLP solvers, this can be done easily.

### A GRID-ENABLED HEURISTIC THAT USES MULTIVARIATE NORMAL SAMPLING

We now consider an improvement on this basic heuristic which lends itself nicely to parallel processing on the grid. Rather than just taking $L = C$, let us take a multivariate normal (MVN) sample around the customer sites to generate a candidate set $L$. This can then be repeated with different seeds for the random number generator. Then, for each candidate set of facilities, we can solve the $p$-median problem and post process the solution concurrently on the grid.

Fortunately, SAS provides all the tools necessary to create such an experiment. The SAS/OR IML procedure provides the Cholesky decomposition needed to produce the MVN sample. A macro to do this is provided on the SAS support website here: `http://support.sas.com/kb/25/addl/fusion25008_1_mvn.sas.txt`.

The following section provides more information about the tools used for putting the complete algorithm on the grid.

## GRID COMPUTING

SAS has had the ability to distribute SAS processing since the introduction of its SAS/CONNECT product. SAS/CONNECT enables the SAS programmer to connect to other computers that have SAS/CONNECT installed and send SAS code to that remote computer for processing. SAS/CONNECT also provides facilities to move data to and from the remote computer.

## SAS GRID MANAGER

Before SAS Grid Manager, SAS/CONNECT required either a telnet daemon or a SAS/CONNECT spawner to be running on the remote computer in order to start the remote SAS/CONNECT server. It also required the name of the remote computer be stored in the code or in a data set so that the code knew which machine to connect to.

SAS Grid Manager enables the remote SAS session to be started by grid middleware without having to know the name of the remote computer. More importantly, the grid middleware can select the least utilized computer in the grid to better balance the processing load. As more computing resources are required, more computers can be added to the grid without the need to change any SAS code.

SAS Grid Manager includes Platform Suite for SAS which contains Platform Computing's Load Sharing Facility (LSF) grid middleware.

## GRID MANAGER FOR MULTISOURCE WEBER PROBLEM

The motivation to use the grid to help with the multisource Weber problem is that each sample from the multivariate normal distribution provides an alternative approximation to the original problem. Each of these cases can be solved completely independently. Using the grid, SAS/OR concurrently computes the optimal solution for all samples. In the end, the least cost solution is selected.

To facilitate this idea, code was written to create a parameter file that has a record for each run that was desired. The parameter data set has variables for the name of the customer location file, the number of facilities to build, a random number generator seed, the solver to use (NLPC, SQP, IPNLP, MILP) and the maximum time to run.

Now the task is to get grid nodes to process the Weber code by using the designated record out of the parameter data set. Since managing connections to remote computers and submissions of work that complete asynchronously on each remote server can be tricky and time-consuming, the new grid-enabled version of %Distribute macro was used. [The original %Distribute macro is documented here: http://support.sas.com/rnd/scalability/papers/distConnect0401.pdf and the grid-enabled version is in the Grid Toolbox found here: http://support.sas.com/rnd/scalability/grid/download.html].

The %Distribute macro relies on a macro called %RInit to do all of its work. %RInit contains two macros, %FirstRSub and %TaskRSub, which are instantiated on every connected grid node. The %FirstRSub macro is run when the grid node is first connected to do any per-node initialization. The %TaskRSub macro is run every time there is work to do and a grid node is idle. These two macros might use some predefined macro variables, such as the following, to determine what work to perform:

- Rem_Host—name of this host
- Rem_iHost—index of this host out of all hosts
- Rem_NIterAll—total number of iterations to process
- Rem_NIter— maximum number of iterations per chunk
- Rem_Seed—random seed, different for each chunk
- Rem_JobIters—number of iterations in this chunk
- Rem_JobID—task number

Rem_Seed, Rem_JobIters, and Rem_JobID change for each run of %TaskRSub. In the Weber problem, %FirstRSub sets up file references for input and output and downloads the parameter data set. When %TaskRSub was called, it used the RemJobID macro variable value as the record number to read the execution parameters out of the parameter data set for this run of the Weber problem.

To determine how many tasks to run, %Distribute looks for two macro variables to be initialized: NIter and NIterAll. NIterAll is the total number of iterations required for the problem, and NIter is the number of iterations each task should process. In the Weber problem, NIterAll is set to the number of records in the parameters data set and NIter is set to 1 so that each task would process only one record out of the parameter data set.

The grid-enabled version of %Distribute requires access to information about the grid. Since this information is stored in a SAS metadata server in a SAS Application Server's Logical Grid Server definition, %Distribute needs to know

the SAS Application Server's name. The %Distribute macro assumes the name will be set in a macro variable called 'GridAppServer.' To prevent the SAS session from prompting for information about the SAS metadata server, the SAS options METASERVER, METAPORT, METAUSER, and METAPASS can be set in the code as well.

So, the process to solve the Weber problem consists of the following tasks:

1. Define the %RInit macro and the macros inside of it (%FirstRSub and %TaskRSub):
   a) The %FirstRSub macro creates file references to input and output locations, downloads the parameter data set, and instantiates the Weber macro that processes the parameters.
   b) The %TaskRSub macro reads the proper parameter data set record specified by Rem_JobID and calls the weber macro with the retrieved parameters.
2. Create the parameter data set that contained a record for the specified customer information, solver, maximum run time, and number of factories. Multiple records were created for the MILP solver for each iteration requested.
3. Set up the macro variables needed for the grid version of %Distribute including:
   a) NIterAll
   b) NIter
   c) GridAppServer
4. Invoke the %Signon macro to begin the process of connecting to grid nodes.
5. Invoke the %Distribute macro to process the data set.
6. Invoke the %Signoff macro to disconnect from the grid nodes and free them up for other grid jobs.
7. Process the results from the grid nodes.

## COMPUTATIONAL RESULTS

### TEST CASES

In order to test the performance of the heuristic algorithm, we chose a test suite that is well-known from the literature on the Weber problem. Information about the test set can be found here (Taillard 2007).

The hardware used was provided by the SAS/OR Research and Development department. It consists of 20 Dell x64 blades, each consisting of 2 quad core Intel Xeon CPUs at 2.33Ghz and 16.0GB of RAM. This gave us 160 cores to run processes, linked together by the SAS Grid Manager framework.

In Table 1 and Table 2 we show results for the well-known example *bon287* with different values of $m$. For the results in Table 1 we use the nonlinear formulation presented in Section "MULTISOURCE WEBER PROBLEM" on page 2. In this experiment, each problem was given a maximum time of $3600$ seconds. In the second column, we give the best-known objective value given in the literature. The third column has the objective reached by SQP. Finally, in the last column we calculate the gap from the best-known solution. As suggested in Section "MULTISOURCE WEBER PROBLEM" on page 2 the local optima obtained by a direct solve can produce low quality solutions. For the cases $m = 2$ and $m = 3$ the solutions are quite good. However, for the other cases, SQP gets stuck in a suboptimal solution which is very far from the best-known solution.

**Table 1** Multisource Weber Problem *bon287* with SQP

| m | Best-Known Objective | SQP Objective | Time | Status | Gap |
|---|---|---|---|---|---|
| 2 | 14,427.59 | 14,482.07 | 50 | Optimal | 0.38% |
| 3 | 12,095.44 | 12,096.41 | 3,600 | MaxTime | 0.01% |
| 10 | 6,705.04 | 20,653.74 | 3,600 | MaxTime | 67.54% |
| 15 | 5,224.70 | 17,698.85 | 3,600 | MaxTime | 70.48% |
| 30 | 2,716.91 | 51,131.67 | 3,600 | MaxTime | 94.69% |

As described in Section "HEURISTIC METHODS" on page 4 we attempt to improve on the results of the nonlinear solvers by using heuristic methods. In Table 2 we show the results of using the set of customers as the set of candidate

facilities in the $p$-median formulation, i.e. $L = C$. These results are in the first section of the table and entitled *MILP (L=C)*. It is interesting to compare the gaps obtained by SQP and the heuristic method that uses MILP. In all cases, the heuristic performs extremely well reducing the gap to less than $1\%$. In the second section of the table we show the improved results of running the post-processing step described in Section "POST PROCESSING" on page 5. These results are entitled *MILP+ (L=C)*. Here we can see that the post-processing step improved the gap in almost every case with a negligible increase in CPU time.

**Table 2** Multisource Weber Problem *bon287* with MILP Heuristic

| m | Best-Known Objective | MILP (L=C) Objective | Time | Gap | MILP+ (L=C) Objective | Time | Gap |
|---|---|---|---|---|---|---|---|
| 2 | 14,427.59 | 14,522.87 | 75 | 0.66% | 14,490.92 | 75 | 0.44% |
| 3 | 12,095.44 | 12,101.47 | 62 | 0.05% | 12,101.47 | 62 | 0.05% |
| 10 | 6,705.04 | 6,757.13 | 50 | 0.77% | 6,745.96 | 51 | 0.61% |
| 15 | 5,224.70 | 5,237.04 | 25 | 0.24% | 5,235.19 | 29 | 0.20% |
| 30 | 2,716.91 | 2,723.51 | 22 | 0.24% | 2,719.07 | 28 | 0.08% |

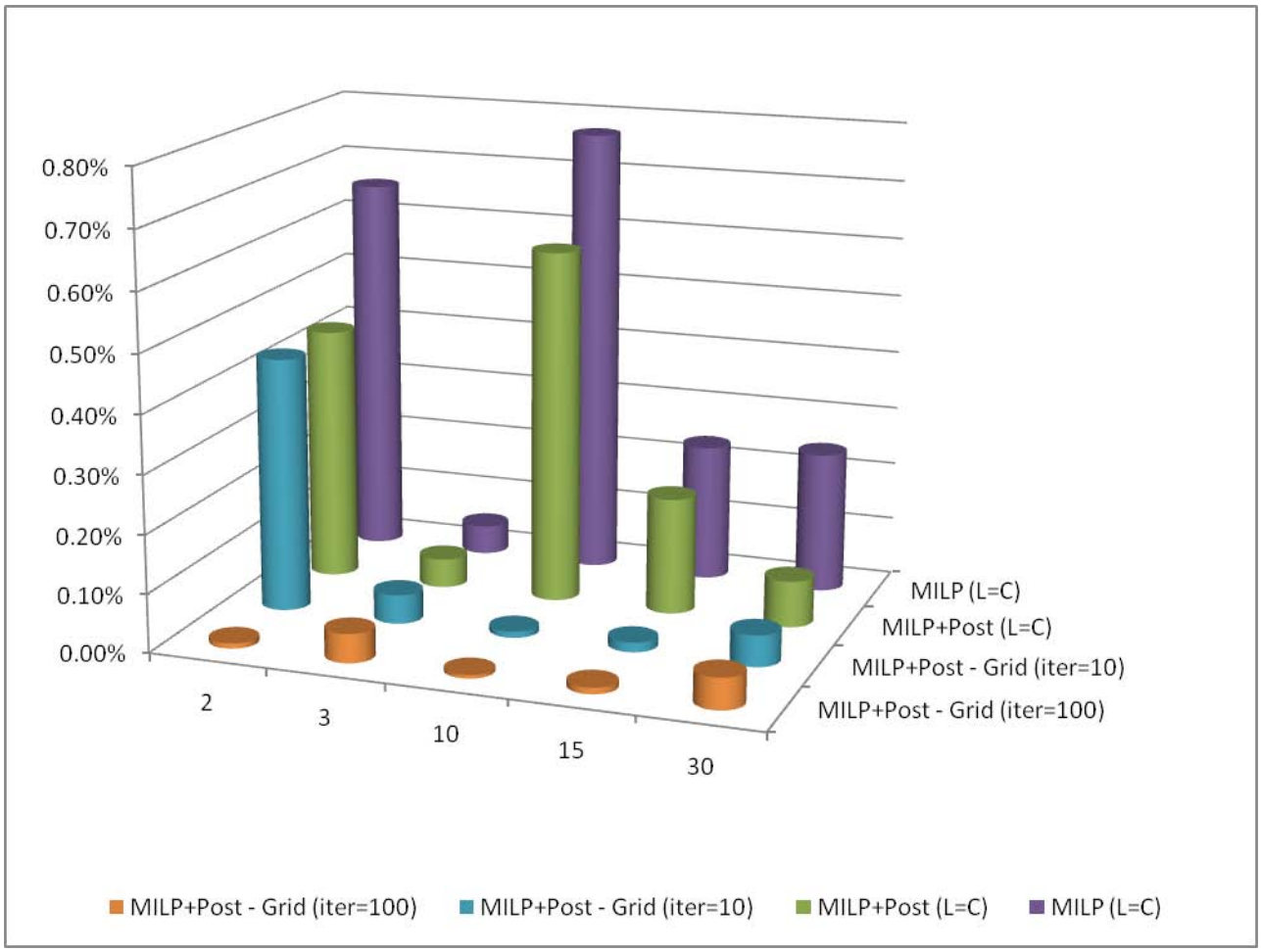
In Table 3 we now show the results of running the experiment on the grid as described in "A GRID-ENABLED HEURISTIC THAT USES MULTIVARIATE NORMAL SAMPLING" on page 5. In the first section of the table, entitled *MILP+ (Grid10)*, we ran the MILP heuristic and post-processing step with 10 multivariate normal samples for candidate facilities. In the second section, entitled *MILP+ (Grid100)*, we used 100 samples. In this table *RTime* now represents *real time* since we are running the computations concurrently. The results show that as we increase the number of samples from 1 to 10 to 100, we get closer to solving each case to full optimality. In fact, after 100 samples we are now within $0.05\%$ gap in all cases. Since the computation is taking place concurrently, the time cost is minimal.

**Table 3** Multisource Weber Problem *bon287* with MILP on Grid

| m | Best-Known Objective | MILP+ (Grid10) Objective | RTime | Gap | MILP+ (Grid100) Objective | RTime | Gap |
|---|---|---|---|---|---|---|---|
| 2 | 14,427.59 | 14,490.92 | 169 | 0.44% | 14,428.87 | 226 | 0.01% |
| 3 | 12,095.44 | 12,101.47 | 124 | 0.05% | 12,101.47 | 188 | 0.05% |
| 10 | 6,705.04 | 6,705.74 | 128 | 0.01% | 6,705.44 | 180 | 0.01% |
| 15 | 5,224.70 | 5,225.56 | 90 | 0.02% | 5,225.28 | 153 | 0.01% |
| 30 | 2,716.91 | 2,718.37 | 88 | 0.05% | 2,718.37 | 154 | 0.05% |

In Figure 4 we illustrate the results in the previous tables to show the iterative improvement obtained on the gap.

**Figure 4** Optimality Gap for Multisource Weber Problem *bon287* with MILP on Grid




## REFERENCES

Fishman, G. (1995), *Monte Carlo: Concepts, Algorithms, and Applications*, New York: Springer Verlag.

Garey, M. R. and Johnson, D. S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: W. H. Freeman and Company.

Hansen, P., Mladenovič, N., and Taillard, E. (1998), "Heuristic Solution of the Multisource Weber Problem as a $p$-Median Problem," *Operations Research Letters*, 22, 55–62.

Horst, R. and Tuy, H. (1993), *Global Optimization*, New York: Springer-Verlag.

Levine, D. (1994), *A Parallel Genetic Algorithm for the Set Partitioning Problem*, Ph.D. thesis, Illinois Institute of Technology, Chicago, IL.

Ralphs, T. and Galati, M. (2005), "Decomposition in Integer Programming," in J. Karlof, ed., *Integer Programming: Theory and Practice*, The Pennsylvania State University: CRC Press.

Robinson, S. (2004), *Simulation—The Practice of Model Development and Use*, John Wiley & Sons.

S. Eilon, C. W.-G. and Christofides, N. (1969), "Distribution Management: Mathematical Modeling and Practical Analysis," *Operational Research Quarterly*, 20, 37–53.

Taillard, E. (2007), "Location Problems," <http://ina2.eivd.ch/Collaborateurs/etd/problemes.dir/location.html> (February 4, 2008).

## CONTACT INFORMATION

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

Matthew Galati
SAS Institute Inc.
Philadelphia Regional Office
Suite 201, 1400 Morris Drive
Wayne, PA 19087
Phone: 610-640-1324, x1431
Fax: 610-640-1488
Email: matthew.galati@sas.com