Paper 200-2008

Using SAS/OR® to Improve Emergency Response Times with Better Ambulance Placement

William Storti, San Francisco Fire Department

Jodi Blomberg, SAS Institute

ABSTRACT

Using the new SAS/OR® procedures, SAS® software and the San Francisco Fire Department analyzed emergency response operations to optimize ambulance locations and relate that information into dispatch procedures to decrease response times. The first model was created as a planning model for initial placement of medical units over the city of San Francisco. A second model was developed to determine how to best place remaining medical units around the city when some number of medical units are already out on an emergency.

INTRODUCTION

In the city of San Francisco, ambulances and fire stations are co-located. There are 41 fire stations in the city and 19 roving medical units that can be assigned to any of the 41 stations. When an emergency call comes in, the closest unit is sent out on the call. Response time reduction is key to saving lives and improving outcomes. The focus of this project was to reduce response time.

A reduction in response time could be achieved in many ways. As an optimization problem, we focused on minimizing the distance between available units and emergency calls. The underlying idea is that a reduction in distance will produce a reduction in response times. This requires a quantifiable definition of the distance between calls and units. Ideally, one could consider the distance between individual calls and a response unit by calculating the distance between every possible call location and possible response unit location. However, this is not a good practical solution because calls can come from literally anywhere in the city. Instead, the city is divided into 174 areas that have approximately equal population and call volume. Subsequently, some areas are very small geographically and some are much larger. For example, the area that covers Golden Gate Park is geographically much larger than the area for small sections of downtown San Francisco. Distance between two points is then measured in terms of the average time to drive between a center point of the area and a fire station. This *distance* definition does not allow us to model differences in traffic due to time of day or other factors but keeps our model and data simple enough for a quick solution time.

Now that the problem of distance is well defined, we need to determine how to go about solving such a problem. In the theory of operations research, this problem is a *p-center* problem. A p-center problem consists of locating p facilities and assigning clients to them in order to minimize the maximum distance between a client and the facility to which the client is allocated. In our case, we are assigning city areas to medical units in order to minimize the distance between emergency calls and the medic units. This requires us to also assign medical units to fire stations, but we simply assign them to the ones that are providing coverage. To minimize distance, you can minimize total distance, average distance, or just minimize the maximum distance. We have chosen to minimize the maximum distance. This approach ensures that all distances will be at or below a specific maximum distance, and it aligns with our goal of response times under a specific threshold.

The remainder of this paper will discuss the formulation of the problem, solution, and practical challenges to implementation.

THE PLANNING PROBLEM

We call the initial problem formulation the *planning problem*. This problem formulation is designed to answer the question: “At a point in time with no ambulances currently in use, at which fire stations should the medical units be stationed to minimize the distance between the 174 areas of the city and the medical units?” This problem addresses where the fire department should plan to station ambulances.

PROBLEM FORMULATION

This section describes the problem formulation of the operations research problem. We include both the equations and *plain English* descriptions. In order to formulate a theoretic basis for a solution, we must formally define what we know and what we hope to determine. We know the locations of stations where ambulances can sit to wait for calls and travel time between any station location and city area. We are looking for how to assign city areas to station location and medic units to station location.

We set up the following indices for the sets that we are going to need.

Index	Set	Description
i	I	set of all fire station locations
j	J	set of all city areas
m	I_m	set of locations that must have a medic unit $\{ i \in I_m \} \subseteq I$

We use the following names and indices to describe the data that we are going to put into the system. This data will be supplied as SAS data sets.

Parameter	Parameter Description
LocationsOpen	number of available medical units in the system
S_i	supply of responses in location i
D_j	demand for responses in area j
Dis_{ij}	distance from location i to area j
defaultOpen	list of locations that must be open ($m \in I_m$)

This describes what we already know. Now we need to set up terms to describe what we are looking to solve. These are commonly known as *decision variables*.

Variable	Description
X_{ij}	= 1 if location i covers area j, 0 otherwise
$Open_i$	= 1 if location i has medic unit, 0 otherwise
MaxDist	= max distance between a location and any of its covered areas

All of the terms and indices used in the following problem formulation are now defined.

Objective Function

The objective function states the goal of the optimization model. In our case, the goal is: Minimize the maximum distance between locations and areas. In our notation, this is:

Min MaxDist

Constraints

The minimization problem has the following constraints.

Max Distance Definition: Define the largest distance

(1) $MaxDist \geq Dis_{ij} X_{ij}$ for all $i \in I$, $j \in J$

Staffed Constraint: Total locations opened must equal the number available to open

(2) $\sum_{i \in I} Open_i = LocationsOpen$

Service Constraint: Locations that cover an area must be open

(3) $X_{ij} \leq Open_i$ for all $i \in I$, $j \in J$

Cover Constraint: Sum of coverage must meet demand. *Demand* here means that each area requires a response. That is, each area must be covered by a medical unit, and any solution provides coverage to all city areas.

(4) $\sum_{i \in I} X_{ij} = D_j$ for all $j \in J$

Supply Constraint: Sum of coverage must not exceed supply. "Supply" here means that each location can provide coverage for some number of areas and coverage only occurs when a location has a medical unit.

(5) $\sum_{j \in J} X_{ij} \leq S_i \ Open_i$ for all $i \in I$

Default Open Constraint: Forces locations in I_m to be open. This constraint is currently unused but provides a means for forcing certain locations to house a medical unit.

(6) $Open_i = 1$ for all $i \in I_m$

The only possible values for X_{ij} and $Open_i$ are 0 and 1. A station cannot be *half open*; it either is open or it is not. That is, X_{ij} is binary integer and $Open_i$ is binary integer. Due to these bounds on the variables, this problem is known as an *integer* programming *problem*.

THE REAL TIME PROBLEM

The planning problem glosses over one of the realities faced by the San Francisco Fire Department. In only very rare instances would all available medical units be sitting and waiting for calls. Most of the time, at least one medical unit is already out on the call. So a better question to ask might be: "If some number of ambulances are out on calls, how should I reposition the remaining medical units to minimize the distance between calls and units?" We call this the *real-time problem*. The real-time problem formulation is the same as the planning problem with the addition of a few sets and a constraint.

We define two new indices *f* and *r* and two new sets: I_f and I_r.
I_f is the set of locations for which all but some number x will be open. I_r contains the remaining set of locations not in I_m or I_f. X of these locations are opened. In set terminology, these sets must have the following relationship to make our problem solvable: $I_f \cap I_r = \emptyset$ and $I_r \cap I_m = \emptyset$ and $I_f \cap I_m = \emptyset$ and $I_f \cup I_r \cup I_m = I$. That is, I_f and I_r cannot have any locations in common, nor can they have any locations in common with I_m. All locations must be represented in one of the three sets.

We also must add one constraint which allows that all but *k* locations will come from the fixed location set. This means that the model can reassign only *k* medical units. The remaining units must stay where they are.

Fixed Location Constraint: Requires that all but *k* location(s) must come from the fixed location set

(7) $\sum_{i \in I_f} Open_i + k = LocationsOpen$

One might wonder why we didn't return to the planning problem and just rerun an optimal configuration for a reduced set of ambulances. The reason is entirely practical. It is not reasonable to ask medical units to move constantly.

Keeping medical units stationary when possible allows crews on the medic units to eat and rest, which are clearly important functions.

OPTIMIZATION SOLUTION

Now that the problems have been defined, they can be solved with SAS/OR® software. First, we write the problem into the SAS OPTMODEL language and then we submit it to the SAS/OR® solvers.

SAS OPTMODEL

Using SAS OPTMODEL, we need write the equations described previously into SAS code. We can write the SAS code to separate the model from the data. All of the OPTMODEL code is not included here but we outline a few constraints to give some examples of how the problem formulations become code.

Consider the service constraint: Locations that cover an area must be open.

(3) $X_{ij} \leq Open_i$ for all $i \in I$, $j \in J$

The following code defines the appropriate sets and the service constraint.

```
proc optmodel presolver=0;
/* Define Data */
      set <string> LOCATIONS;
      number Supply { LOCATIONS };
      read data fire.supply into LOCATIONS =[location] supply;
      set <string> AREAS;
      number Demand{AREAS};
      read data fire.demand into AREAS=[area] demand;

      num distance {LOCATIONS, AREAS};
      set <string, string> StatAreaIndex;
      read data distance2 into StatAreaIndex = [location area] distance;

* Define Variables;
      var X { LOCATIONS, AREAS} binary;
      var Open { LOCATIONS } binary;

* Define Constraints;

* Service Constraint;
      con service_con {i in LOCATIONS, j in AREAS}:
      X[i,j] <= Open[i];
```

The line: `set <string> LOCATIONS;` simply defines that we are creating a set called locations that is text. The line `number Supply {LOCATIONS}` defines a number called supply, which is indexed by locations. All of our sets are set up similarly.

Now that we have a set and a number defined, we can read the data into them using a read statement.

```
read data fire.supply into LOCATIONS=[location] supply;
```

We designate that the data set fire.supply is our data source. Because our number supply is dependent on set locations, we read the variable location (lower case) into the set LOCATIONS (upper case). We then read the supply column (name of column) into the number supply.

In the Define Variables section, two variables are defined: x and open. Both are indexed by the sets between the brackets { }. Both are binary, so the word binary must follow their declaration.

```
* Define Variables;
```

```
var X {LOCATIONS, AREAS} binary;
var Open {LOCATIONS} binary;
```

In the Define Constraints section, the service constraint is constructed. All constraint definitions start with "con" and then the name of the constraint followed by a colon. If the constraint should be created for each instance in a set (for example, for each location and area), then we need to define that $X_{ij} \leq Open_i$. Before the colon, the index for which a constraint should be created is defined. This is equivalent to a "for all $i \in I$ $j \in J$" following a constraint in the algebraic formulation. After the colon, the constraint itself is defined in terms of the variables and sets already defined.

FINDING A SOLUTION

Because this is an integer programming problem, we need to use an integer programming solver. The SAS solver we use is the MILP- Mixed Integer Linear Programming solver. The solution time for this is too long to solve in a straightforward manner. Other integer programming solvers used by the San Francisco Fire Department also resulted in very long solution times of twenty hours or more. The solution to this problem is not really useful unless it can be solved quickly and passed to the dispatchers.

We then turned to SAS Research and Development to find a faster way to arrive at a solution. Fortunately, a very fast optimal solution – less than one minute – can be achieved with a binary search. The key to understanding how this works is to remember that we are looking to minimize the maximum distance between a city area and a medic unit. Thus, the solution to our problem lies somewhere in the list of distances and we just need to go look for it.

The following outlines our search for the minimal maximum distance and we use the MILP solver to help us.

(1) Set a lowerbound for maxdistance
lowerbound = max(j in AREAS) min(i in Stations) distance (i,j)

(2) Set dummy objective function. (for example, min 0)

(3) Solve as a series of feasibility problems. Since the objective maxdist of the p-center problem is equal to one of the distance[i,j]; sort these distances and then perform a binary search, at each stage solving a feasibility problem where arcs with larger distance[i,j] are dropped.

Follow these steps:

a. Create an ordered array containing the distances.

b. Choose the distance in the middle of the array (call it *m*).

c. Create a constraint that sets all $X_{ij} = 0$ for any distance greater than *m*.

d. If a feasible solution to this problem exists, then the minimum maxdist is *m* or smaller, so you can start looking at a new *m*'< *m* and repeat from step c.

If a feasible solution does not exist, then the minimum maxdist value must be larger than *m* and you can start looking at new *m*'> *m* and repeat from step c.

Steps a – d are run iteratively until a feasible solution with the lowest maxdist value is found.

Since there can be many solutions that have this minimum maximum distance, we then solve the problem again to minimize total distance holding the maximum distance to the level found in step 3.

Using this method to solve either problem set results in a very fast solution time of under one minute.

IMPLEMENTING THE SOLUTION

The results of the model need to be communicated to the dispatchers. San Francisco Fire Department uses a computerized dispatch system that visually tracks movement of ambulances using GPS. The system does not control

the movement of medical units; dispatchers tell medical units when to move stations. While a wide variety of data can be read out of the model solution into SAS data sets, the dispatchers need very limited information. They need information only of the following form: “Move ambulance x from station 1 to station 2.”

In fact, in the proposed implementation, dispatchers get a suggestion on where to put remaining ambulances. They can override the solution if they think they know a better solution due to their experience with traffic patterns and call types. Because calls come in constantly, the dispatcher needs a consistently updated solution. This drives our need for a very quick solution time of under one minute.

CONCLUSION

When we implemented an optimization model, practical considerations played a large part in driving both the problem formulation as well as the solution method. While optimization is the goal, the requirements for a fast solution time and the needs of the crew on the medical units and San Francisco Fire Department are equally as important to the end solution.

CONTACT INFORMATION

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

Jodi Blomberg
SAS Institute
6501 S Fiddler’s Green Circle
Suite 600
Greenwood Village, CO 80111
303-282-6605
jodi.blomberg@sas.com

William Storti
San Francisco Fire Department
698 Second Street
San Francisco, CA 94107
(415) 558-3200
Bill.Storti@SFGOV.ORG