Summer 2015

The Facility Location Problem

Meghan E. Csoke
Governors State University

The Facility Location Problem

By

Meghan E. Csoke
B.S., Iowa State University, 2004
B.A., Governors State University, 2011

Thesis

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science
With a Major in Mathematics

Governors State University
University Park, IL 60466

2015

Abstract

The purpose of this study was to analyze the location of an emergency facility location within a town based on given information from the village and to use the results to determine the optimal location for an emergency facility. A model of the problem was developed using a spreadsheet and computer program to record and analyze the optimal response time based on different locations of emergency facilities. Assumptions were made to create situations easily computed through spreadsheet and computer programs. Once calculated, information was used to create a framework of demand density across a gridded map. Once the computer program was updated to use the large amount of data, results were obtained. Based on data and modeling, the current location of emergency facility was not located in the most opportune locations and another location was deemed better suited for serving the community.

1 Introduction

The facility location problem is an optimization problem that appears in many disciplines and whose methods of solution can be applied to a vast range of initial problems. There are two main categories of facilities: service based vs. industry based. The goal is to focus on the service based aspect of this problem. Service facilities can be, but are not limited to, police stations, fire stations, hospitals, schools, libraries, ambulance depots, emergency care centers, etc. These facilities are stationary and provide or assist in providing a service to a community. We will be looking into seeking the optimal locations of facilities to optimize response time to better serve a given community. The location of emergency facilities significantly affects the safety and well being of the community as mentioned by Caccetta and Dzator [1]. Response time has been dictated by the National Fire Protection Agency (NFPA) in their code book [2]. NFPA publishes 300 codes and standards that are designed to minimize the risk and effects of fire by establishing criteria for building, processing, design, service, and installation in the United States, as well as many other countries.

The safety and well being of the community depends directly or indirectly on the response time of the emergency facilities [1]. The minimization of the response time measures the performance of emergency facilities and the performance of these facilities can be improved by either improving the existing location of emergency facilities or increasing the number of facilities [1]. Increasing the number of facilities can be a challenge based on allocated space and city funding. Due to these factors, it is crucial to locate a facility effectively and efficiently. The NFPA has stated in their standard 1710 (standard for the organization and deployment of fire suppression operations, emergency medical operations, and special operations to the public by career fire departments) that the travel time for a fire engine or ambulance should be 240 seconds or less and

480 seconds or less for the arrival of all services for a full alarm or advanced life support (ALS) unit [2]. In this paper, data was examined from the Oak Lawn Police and Fire Department to verify their locations are the optimal locations for police and fire. The data was also examined to determine whether or not adding more facility locations should be considered.

An important way to measure the efficiency and effectiveness of an emergency facility is by evaluating the average distance between the customers and the facilities [1]. A decrease in response time will occur when the average distance between the facility location and emergency location decreases. This is known as the p-median problem. According to Hakimi [3], the p-median problem is that of locating p facilities to minimize the demand weighted average distance between demand nodes and the nearest of the selected facilities and was originally introduced in 1964. Hakimi [4] showed that one optimal solution to the p-median problem has locations only on nodes which will be the type of solution this paper will focus on.

Since the p-median problem is a computationally difficult problem to solve (the problem is NP-hard on general graphs) [5, 6], heuristic models were explored. NP can be defined as a class of computational problems for which a given solution can be verified as a solution in polynomial time by a deterministic Turing machine. Being NP-hard is defined as a class of problems which are at least as hard as the hardest problems in NP. Classical methods compute all possible outcomes and record the best possible solution. Heuristics use methods to approximate the best possible answer in a shorter amount of time. There are many heuristic approaches to solving the p-median problem including: genetic algorithms, simulated annealing, tabu search, node partitioning, node insertion, node substitution, and various hybrids [7, 8, 9, 10]. A disadvantage of using heuristics to solve a p-median problem is that there is no way to verify if the given solution is in fact the best solution unless you have already solved the problem by using a classical method. Tabu search will be used in the modeling portion and is actually considered a meta heuristic. Heuristics find 'good' solutions on large-size problem instances and they allow acceptable performance at acceptable costs in a wide range of problems. Heuristics do not have an approximation guarantee on the obtained solutions. They are tailored and designed to solve a specific problem or/and instance. Meta-heuristics are general-purpose algorithms that can be applied to solve almost any optimization problem. They may be viewed as upper level general methodologies that can be used as a guiding strategy in designing underlying heuristics [11].

2 Modeling Competition Problem

A simple case that was introduced during the 1986 Mathematical Competition in Modeling and presented by the Department of Mathematical Sciences at Salisbury State College in Salisbury, Maryland [12] was considered. Over a weekend in February, 1986, three-student teams were presented with a packet and a choice of two problems to solve. The first asked for construction of a contour based on given hydro-graphic data, and the second asked for an optimal location of two emergency facilities in a small town. All teams were free to use computers and libraries. The second presented problem is below [12, pp1-2].

> The township of Rio Rancho has hitherto not had its own emergency facilities. It has secured funds to erect two emergency facilities in 1986, each of which will combine ambulance, fire, and police services. Figure 1 indicates the demand, or number of emergencies per square block, for 1985. The L region in the north is an obstacle, whereas the rectangle in the south is a park with a shallow pond. It takes an emergency vehicle an average of 15 seconds to go one block in the

N–S direction and 20 seconds in the E–W direction. The task was to locate the two facilities so as to minimize the total response time. It was assumed that the demand is concentrated at the center of the block and that the facilities will be located on corners. Assume that the demand is uniformly distributed on the streets bordering each block and that the facilities may be located anywhere on the streets.

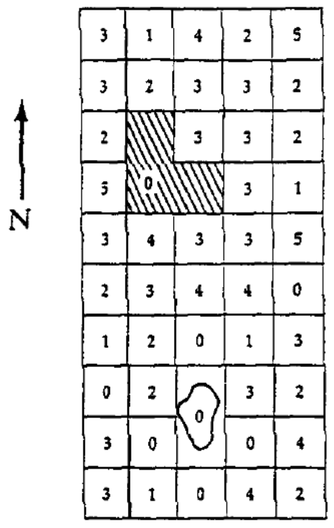


Figure 1: A map of Rio Rancho, with number of emergencies in 1985 indicated for each block [12].

2.1 Assumptions

To solve the Rio Rancho problem, assumptions needed to be made based on the given information. These assumptions enabled the coding to work with fewer complications.

- A dispatcher will send out an emergency vehicle from the facility closest to the emergency site.
- There will always be an emergency vehicle available at the emergency

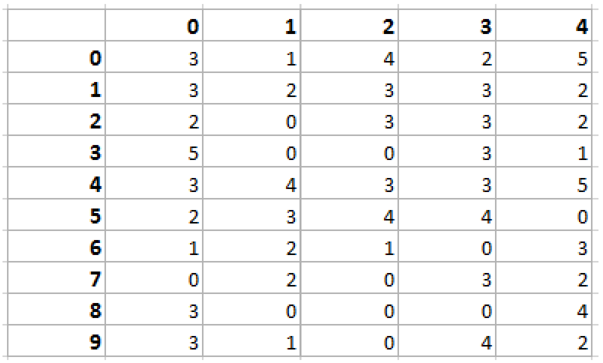
facility closest to the site.

- The optimal location of an emergency facility will be at an intersection which enables the emergency vehicle to begin its path in either direction.
- Emergency vehicles must drive on existing streets.
- Emergency vehicles always choose the shortest route distance wise.
- An emergency vehicle is called only from a facilities location, it will not travel from one emergency to the next.
- Emergency facility locations can be located on any corner (nodes or points) with in a specified block number.

2.2 Solution

For this solution, assume that the demand is concentrated at the center of the block and that the facilities will be located on corners. The response time will need to be minimized. This will be done while assuming the emergencies occur at the center of a block and the obstacles shown on the map (Figure 1) are negated. This means the emergency vehicle has arrived once it is at any corner of the block. To try and solve this part of the problem, we can start by modeling a single emergency facility location and minimizing response time through all possible locations. This was done so that there was a basic code structure to work with and then expand further for future models. The model to solve this situation was created using Visual Basic (VBA) in Excel with multiple macros for each part.

First, a map of Rio Rancho was created in Excel with a 10 by 5 grid where each cell represented a block in the town. Values were entered in each remaining cell for demand. The block numbers were represented by their $x-$ and $y-$ coordinates on the 10 by 5 grid. The coordinates started at 0 and ended at 9 in the north-south direction and started at 0 and ended at 4 in the east-west direction. This grid, Figure 2, was used as a visual reference and guide for the remaining parts of the code.



	0	1	2	3	4
0	3	1	4	2	5
1	3	2	3	3	2
2	2	0	3	3	2
3	5	0	0	3	1
4	3	4	3	3	5
5	2	3	4	4	0
6	1	2	1	0	3
7	0	2	0	3	2
8	3	0	0	0	4
9	3	1	0	4	2

Figure 2: Rio Rancho Block Grid with Demand

The first challenge of this problem was creating a list of all possible combination of points by coding in VBA. The combinations represent the emergency facility location and the actual emergency. For example, if the facility was located at any corner of block (0,0), it was necessary to determine the distance from (0,0) to all other possible blocks (50 of them in this case) in the town. Since there are 50 nodes there will then be 2500 different combinations of where the emergency facility can be located and where all possible emergencies will be located. The code used to create this list of point combinations used can be found in Appendix A.

A separate macro was created in VBA to determine the response time between each pair of points in the list of 2500 combinations (Appendix B). A simple formula for rectilinear distances was implemented and adjusted with the given time constraints as stated in the original problem and can be seen in Listing 1. Lines 5-10 are declaring variables, lines 14-15 are telling the program where to get the information, lines 19-22 are telling the program where the defined variables are located and what type of information to take from the location, line 26 is where you can find the formula for rectilinear distances adjusted with the travel times, and lines 30-32 tell the program where to insert the calculated response time and to repeat the process. Emergency vehicles moving in the N-S direction would require 15 seconds to travel each block and vehicles traveling in the E-W direction would require 20 seconds. These times were added into the code as multipliers to the original rectilinear formula. The pair (x_1, y_1) represents the coordinates of the facility and (x_2, y_2) represents the coordinates of an emergency. The time it takes an emergency vehicle to respond from a specific location and travel to a specific emergency site is called the response time. That response time needs to be multiplied by the demand at the emergency location block to determine the total response time (at any given block). This was done in VBA with a separate macro as well and can be seen in Appendix B.

Listing 1: Calculate Distances Code

```
 1  Sub CalculateDistance()
 2
 3  'Declariation of variables and their types
 4
 5  Dim x1 As Double
 6  Dim x2 As Double
 7  Dim y1 As Double
 8  Dim y2 As Double
 9  Dim Distance As Double
10  Dim i As Long
11
12  'extract input from the sheet
13
14  With Sheets("Combination␣of␣Points")
15      For i = 15 To 2514
16
17          'select the cell itself
18
19          x1 = Range("B" & i).Value
20          x2 = Range("D" & i).Value
21          y1 = Range("C" & i).Value
22          y2 = Range("E" & i).Value
23
24          'formula for distance with travel time is (abs(x1 -
                x2) * 20) + (abs(y1 - y2) * 15)
25
```

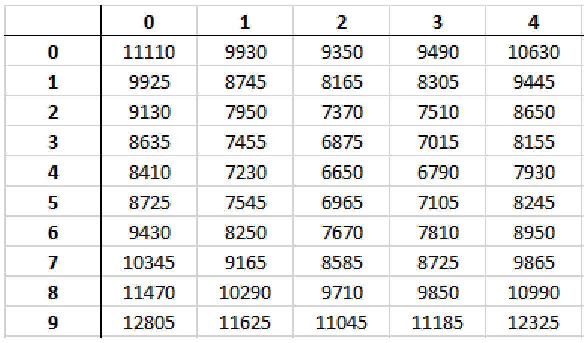
```
        Distance = (Abs(x1 - x2) * 20) + (Abs(y1 - y2) * 15)

        'output response time into specified cells

        Range("G" & i).Value = Distance
    Next i
End With

Beep
```

Next, there needed to be code written to take the list of 2500 possible response times for the Rio Rancho area and covert it back into a visually understandable grid like we started with. Within this grid we also needed to sum the total response time over an entire block based on the location of the emergency facility to show the total response time for the entire town if the emergency facility was located at that specific block. Also included in the code (Appendix C) are the three smallest total response times.



	0	1	2	3	4
0	11110	9930	9350	9490	10630
1	9925	8745	8165	8305	9445
2	9130	7950	7370	7510	8650
3	8635	7455	6875	7015	8155
4	8410	7230	6650	6790	7930
5	8725	7545	6965	7105	8245
6	9430	8250	7670	7810	8950
7	10345	9165	8585	8725	9865
8	11470	10290	9710	9850	10990
9	12805	11625	11045	11185	12325

Figure 3: Gridded Total Response Time

In Figure 3 it can seen where the three best locations for one facility in the Rio Rancho area should be located, in the block (2, 4), (2, 3) and (3, 4). The location with the smallest amount of response time would be at block (2, 4).

2.3 Strengths and Weaknesses of the Model

Finding a solution through the method described in this paper will truly only work in this instance. Due to the complication of coding this in VBA, a general case code could not be completed. Because of this, the original stated problem could not be determined. Locating two emergency facilities would include many if/then statements and looping. The concept can easily be mapped but the act of writing code to complete the process is beyond this author's scope. For example, an if statement testing a current location must sum up all of the travel times to each 50 nodes, save that time in memory, and check the next location. If the new sum is less that the sum in memory, the program needs to forget the old, save the new and continue checking until all 50 locations have been summed and verified for the smallest total response time.

2.4 Conclusion

The solution gained from using this program was consistent with the solutions presented in the research material. For a one facility location problem, it is almost as easy to visually see the location as it is to find the optimal location using data, once a proper program has been created. Considering the solutions the algorithm found, the obstacles do not play an important role in determining the optimal solution. This means there is no reason for Rio Rancho to remove the obstacle or bridge the pond to decrease response time. Since the station is not located in blocks which are blocked by either the L-shaped obstacle or the pond, no extra time need be expended avoiding the obstructions (detour), therefore they have no effect on the optimal solution.

3 Application of Model

After creating a working model in Excel for the Rio Rancho problem, a more organized program needed to be created to apply this method to a real life problem situation. Data was gathered from the Oak Lawn Police Department on police and fire calls with in the Oak Lawn, Illinois town for the year 2014. The goal of applying our method to a real life situation is to see if the one emergency facility the town currently has is the best location and where two facility locations would best be placed. As stated previously in this paper, adding a location can depend a multiple factors, including cost and available land, so the outcome is strictly hypothetical.

3.1 Data Sorting

Included in the data was the location of an emergency, the agency who responded, the time the call was taken, the time the agency arrived, and the amount of time to respond. To determine the demand per block, all calls needed to be mapped and assigned one of the 749 different blocks in Oak Lawn. This was done by using Google Maps [13] and plotting all address calls. Addresses are defined as street intersections and not house numbers. There were a total of 30,111 total calls for police and fire in 2014 sorted into 7,064 for fire and 23,047 for police. Since the addresses were listed by intersections, there were multiple calls for the same intersections. The number of times an address appeared in the data list will now be referenced as the demand. Due to this only 2,230 calls needed to be mapped and sorted to each of the blocks. Once this was completed the total demand per block could be determined by using Excel. The following assumptions were made in order sort the data more efficiently.

- All emergencies located at the current police station will be disregarded and not used for this program. It is most likely that the amount of calls generated at the current facility's location would not occur if the facility was not located there.

- Emergency facilities can be located anywhere with in a block.
- Demand will be centered in each block.
- All calls are located at intersections and need to be assigned a block. Therefore, calls will be assigned to the nearest block to the south, or the nearest block to the east of the intersection.
- If there is no block to the south or east (blocks on the boundary of the city), then the call will be assigned to the nearest block on the north or west.
- All emergency response vehicles will be dispatched from their emergency facility. No responders will go from emergency to emergency.

3.2 Tabu Search

Tabu search is a local search method that moves at each iteration from a solution to its best neighbor even if this causes the objective value to deteriorate. Local (neighborhood) searches take a potential solution to a problem and check its immediate neighbors (solutions that are similar except for one or two minor details) in the hope of finding an improved solution. Tabu search enhances the performance of local search by relaxing its basic rule. At each step, worsening moves can be accepted if no improving move is available. It does what local search methods often do: when you get stuck, you allow a non-improving move in the hopes of getting unstuck. In addition, tabu moves are introduced to discourage the search from coming back to previously visited solutions. Tabu search, in particular, maintains a tabu list. When a non-improving move in introduced, the tabu list ensures we never move to a previously visited state.

Multiple parameters of a tabu search exist and includ: local search procedures, neighborhood structures, aspiration conditions, form of tabu moves, addition of a tabu move, maximize size of tabu list, and the stopping rule [14]. A chief way to exploit memory in tabu search is to classify a subset of the moves in a neighborhood, or search space, as forbidden (or tabu) [9]. A neighborhood is constructed to identify adjacent solutions that can be reached from current solution [15]. The classification of the moves depends on the history of the search, and particularly on the recency or frequency that certain move or solution components, called attributes, have participated in generating past solutions [9]. Tabu restrictions are subject to an important exception. When a tabu move has a sufficiently attractive evaluation where it would result in a solution better than any visited so far, then its tabu classification may be overridden. A condition that allows such an override to occur is called an aspiration criterion [9].

Three strategies to using tabu search include the forbidding strategy (control what enters the tabu list), freeing strategy (control what exits the tabu list and when) and short-term strategy (manage interplay between the forbidding strategy and freeing strategy to select trial solutions). To avoid cycling, solutions similar to recently examined solutions are forbidden, or tabu, for a number of iterations [16]. It uses flexible memory and responsive exploration in guiding the solution process to move from one trial solution to another [17]. By giving recently or frequently (or infrequently) visited solutions a tabu status, so as to discourage their selection in the search process, it guides other searching methods to move away from local optimal solutions.

Tabu search has three major flexible memory components, a short term memory process, an intermediate memory process, and a long term memory process [9]. The short term memory process is based on a set of tabu conditions and aspiration criteria. Through frequency-based memories, tabu search characterizes a subset of potential moves as tabu, or forbidden, to avoid reversal of previously visited solutions. The intermediate memory process is implemented by restricting the search within a set of potentially prosperous solutions to intensify the search. The long term memory process is invoked periodically to lead the search to new regions that might not have been explored as it diversifies the search.

The basic tabu search algorithm [18] is outlined below. Here S is the set of feasible solutions (candidate list of solutions), f is some function to be optimized, i is the current solution, j is the next solution, $i*$ is the best solution found so far, and k is the iteration counter. As soon as any non-improving moves are possible, the risk of repeating moves is present. This is when the use of memory is helpful to forbid moves which might lead to repetition in solutions. If such a memory is introduced, we may consider that the structure of neighborhood of the current solution $N(i)$ will depend upon the iteration k; thus the neighborhood would be referred to as $N(i, k)$. The flow chart of the process can be seen in Figure 4.

1. Chose an initial solution i in S. Set $i* = i$ and $k = 0$.

2. Set $k = k + 1$ and generate a subset $V*$ of solutions in $N(i, k)$ such that either one of the tabu conditions is violated or at least one of the aspiration conditions holds.

3. Choose a best j in $V*$ (with respect to f) and set $i = j$.

4. If $f(i) < f(i*)$ then set $i* = i$.

5. Update tabu and aspiration conditions.

6. If a stopping condition is met then stop. Else go to Step 2.

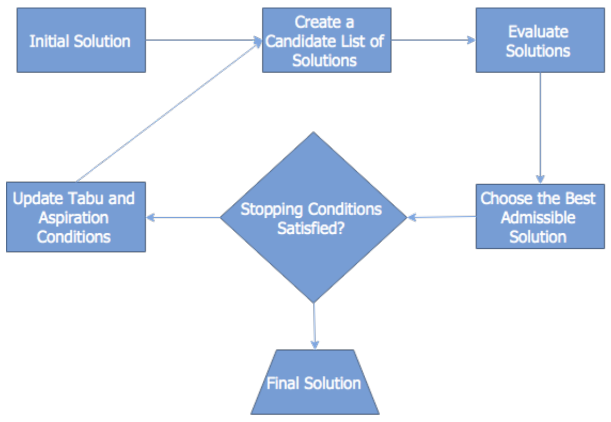

Initial Solution
Create a Candidate List of Solutions
Evaluate Solutions
Choose the Best Admissible Solution
Stopping Conditions Satisfied?
Update Tabu and Aspiration Conditions
Final Solution

Figure 4: Flow Chart of Standard Tabu Search Algorithm

Hillier and Lieberman [14] outlined the tabu search stopping criterion by using a fixed number of iterations, a fixed amount of CPU time, or a fixed number of consecutive iterations without an improvement in the best objective function value. Also, it may stop at any iteration where there are no feasible moves into the local neighborhood of the current trial solution. Some immediate stopping conditions could be the following:

- $N(i, k+1) = 0$. (no feasible solution in the neighborhood of solution i)
- k is larger than the maximum number of iterations allowed.
- The number of iterations since the last improvement of $i*$ is larger than a specified number.
- Evidence can be given than an optimum solution has been obtained.

Various pros and cons are attributed to using the tabu search method. It allows non-improving solution to be accepted in order to escape from a local optimum. The use of tabu list can be applied to both discrete and continuous solution spaces. For larger and more difficult problems (scheduling, quadratic assignment and vehicle routing), tabu search obtains solutions that rival and often surpass the best solutions previously found by other approaches [9]. However, if there are too many parameters to be determined, the number of iterations could be very large and global optimum may not be found, depending on parameter settings.

3.3 The FLP Solver Solution Algorithm

Due to the limitations of the first program created in VBA, a new program needed to be used. The program used for the Oak Lawn data is adapted from the public code for the FLP Solver which was created by Dr. Güneş Erdoğan at the School of Management, University of Southampton [19]. The FLP Solver is an extension of a previously created workbook to help solve the vehicle routing problem. The purpose of the FLP solver workbook is to provide a proof of concept for what can be done when trying to solve the facilities location problem and it was with this in mind, the code provided for the public workbook was adapted to fit the Oak Lawn data.

The field of facilities location problem research mostly focuses on exact algorithms. One of a better known heuristic algorithm to solve this problem is the tabu search [19] and a variant of the tabu search is implemented within the FLP Spreadsheet Solver [19]. An outline of the algorithm is given below.

1. **Initialization:** Initialize the incumbent solution, the best known solution, and the iteration counter $k = 1$. Initialize the tabu list as an empty list. Read the solution on the Solution worksheet into the incumbent solution if a “warm start” is required.

2. **Stopping condition:** If the time limit is exceeded, stop and report the best known solution.

3. **Select move:** Explore all possible relocations of a single location and all

possible exchanges of two locations between facilities. An exchange means an exchange between the emergency location and the emergency to test which location would give us an optimal result (shortest route/time). If the best move results in a solution better than the best known solution, execute it or keep it as the best solution to test other combinations of facilities and emergencies against. Else, choose and execute the best move that does not involve any locations in the tabu list. Add the location(s) in the move which did not get kept as an optimal location to the tabu list.

More specifically, determine the total response time to all locations from a starting location. Next, test another location and calculate the total response time. Those with smaller total response time will be kept, those with greater will be assigned to a tabu list.

4. **Best solution update:** If the incumbent solution is feasible and better than the best known solution, update the best known solution.

5. **Tabu list update:** Remove all locations with a tabu tenure larger than the tabu tenure limit from the tabu list. Go to Step 2. Tabu tenure is defined as the length of time t for which a move is forbidden. If t is too small there is a risk of cycling. If t is too large, it may restrict the list too much.

3.4 New Program - The FLP Solver (Adapted)

The FLP Spreadsheet Solver adopts an incremental flow of information, with subsets of data being kept in separate worksheets, as depicted in Figure 5. Initially, the workbook only contains the worksheet named FLP Solver Console. The remaining worksheets are generated in the sequence denoted by their indexes. Instead of having a wizard interface, which is very easy to use but also very restrictive, the workbook numbers the worksheets in the order of progress.

The parameters related to each worksheet are presented along with their sequence number. Any modifications to the original code will be discussed along with each parameter which are included in the FLP Solver's manual.

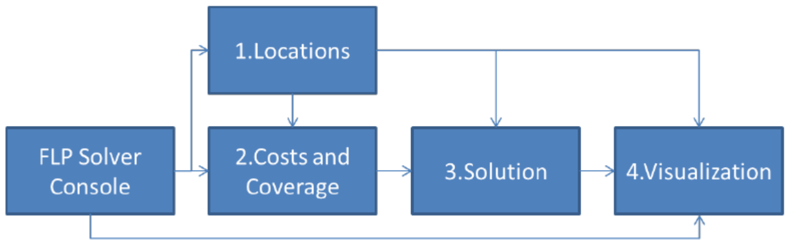


Figure 5: FLP Solver Flow of Information

Bing Maps Key: Having a Bing Maps License is optional. You can still use the workbook without a Bing Maps License. A valid key is required for populating the latitude/longitude, the distances and duration, and for generating visualization of the locations and the routes on a map. You can generate a free key at https://www.bingmapsportal.com/. The key can be copied and pasted into the appropriate cell.

Number of locations: Limited to the interval [10, 200]. This interval was increased to 800 based on the number of possible block locations in Oak Lawn for an emergency facility. The code now has an interval of [10, 800].

Distance computation: This parameter describes how the distances should be populated, if they will be. The options are Manual entry, Euclidean distances, Rounded Euclidean distances, Geodesic approximation, Rectilinear (Manhattan) distances, and Bing Maps. The option Manual entry disables the distance population function. The option Euclidean distances computes the distance between points (x_1, y_1) and (x_2, y_2) as $d_{1,2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, and the results of the formula are not in kilometers but instead are in unit distance.

The option Rounded Euclidean distances uses the Euclidean distance formula and rounds the result to the closest integer, and again the results of the formula are not in kilometers but are measured in unit distance. The option Geodesic approximation uses a spherical approximation for the surface of the Earth, and the results of the formula are in kilometers. This option is useful if you need routes for vessels or aircraft instead of vehicles. The option Rectilinear (Manhattan) distances computes the distance between points (x_1, y_1) and (x_2, y_2) as $d_{1,2} = |x_1 - x_2| + |y_1 - y_2|$, and the results of the formula are not in kilometers but again are measured in unit distance. This does not take into account any obstacles and assumes a complete grid. This option was used to verify another location for the Rio Rancho problem with results discussed later on. The option Bing Maps uses the web service (with the options of avoiding tolls and optimizing for shortest distance), and the results of the formula are in kilometers. This option uses real map data so it will reject routes that pass through a pond or park like in the Rio Rancho problem. This option will be used for the Oak Lawn data.

Bing Maps route type: This parameter describes the type of route returned by Bing Maps. The options are Shortest, Fastest, and Fastest - Real Time Traffic. Shortest option will find the shortest path, which usually goes through city centers, is subject to strict speed limits, and ends up with a very long duration. The recommended option is Fastest. The option Fastest - Real Time Traffic will give you estimates at the time the distances are populated, which may change drastically based on the traffic conditions at that instant. For emergency vehicles, lights and sirens may be applied and are therefore not as strongly inhibited by traffic lights, stop signs and speed limits as normal traffic may be.

Cost per unit distance: The distances are multiplied by this amount to determine the cost of assigning a location to a facility. This relationship is built into the Costs and Coverage worksheet as a formula. There is no cost associated with locating a facility in this problem so the cost portion of the program was altered so it represented demand. Total distances based on time using rectilinear distances and using Bing Maps will be multiplied by the demand (cost) to create a total response time to a block based on the given data.

Costs scaled by demand: Optionally, the assignment cost of a location to a facility may depend on the demand of the location. If the option Yes is selected, the cost formula also involves the demand of the location as a multiplier. In this problem, demand and cost were considered multipliers for the service distance since cost is now representing time.

Service distance limit: Assigning a location to a facility at a distance more than the value of this parameter is prohibited, and a solution involving such an assignment is considered infeasible. This option may be used for solving FLP variants such as distance constrained p-median. This is not necessary in the current problem we are investigating because the facility needs to service the entire area.

Coverage distance limit: Assigning a location to a facility at a distance more than the value of this parameter results in a coverage of 0. The coverage distance limit did not apply here and so it was assigned a very large value so that it is negligible when running the program.

Coverage type: If Step function is selected, 100% of the demand of a location is covered if it is assigned to a facility less than or equal to the coverage distance limit, and none of the demand is covered otherwise. If Linearly decreasing coverage is selected, the percentage of the demand of a location covered by the facility it is assigned to is computed as max $\{0, 1 - $(distance from the facility to the location / Coverage distance limit)$\}$. Step function will be used for the Oak Lawn data.

Number of facilities: This parameter denotes the maximum number of facilities to be located. This may be set to the number of locations if the actual number of facilities located is to be determined by the solution algorithm.

Objective: The options are Minimize total cost, Maximize demand covered, and Minimize maximum service distance. For both the Rio Rancho problem and the Oak Lawn data the minimize total cost option will be selected. This is actually based on demand since there is 0 cost associated with locating a facility.

All facilities must be located?: If Yes is selected, the solver and the feasibility checker require the number of facilities located in a feasible solution to be Number of facilities. If No is selected, a feasible solution may have a smaller number of facilities located.

Visualization background: The options are Bing Maps and Blank. If Bing Maps is selected, the workbook will download the appropriate map containing the coordinates of the locations, and use it as the background image of the scatter chart depicting the routes. This option was eliminated since the scatter plot does not accurately visualize the routes taken by using Bing Maps. Instead the visualization will depict all vehicles traveling using Euclidean distances even if another option for calculating distance was selected.

Location labels: The options are Blank, Location IDs, and Location names. If Location IDs is selected, the ID number of the location will be displayed next to the location on the map. If Location names is selected, the name of the location will be displayed next to the location on the map.

Warm start: If “Yes” is selected, the solution algorithm will attempt to use the solution on the solution worksheet as a starting point.

CPU time limit: This parameter denotes the run time limit of the FLP Spreadsheet Solver. As a general rule, a longer run gives a higher probability of finding a good solution. The FLP Spreadsheet Solver will not stop before completing the first iteration, which may be longer than the time limit provided.

The FLP Solver can be interrupted by pressing the ESC key.

3.5 Results

The FLP Solver was also run for the Rio Rancho problem and was tested under multiple situations to find which option works best when applying the Oak Lawn data. First, the program was run to find the optimal location for one facility to compare with the results from the Rio Rancho VBA code. The results using rectilinear (Manhattan) distances and the objective to minimize total cost gave the results of (2, 4), the same results as the original program. It should be noted that the other two objective options returned the same result as well. This verifies that the FLP solver can determine an optimal location using a tabu search.

Second, the program was run to locate two facility locations using the objective option, maximize demand covered and distances determined using rectilinear (Manhattan) distances. For the Rio Rancho problem there were 50 different possible locations for facilities located anywhere in a given block number (or ordered pair). This means that the emergency facility could be located any where within the given block number. Under these circumstances the program determined the facilities to be located at $(2, 1)$ and $(2, 6)$. These two locations maximize the demand assigned to an emergency facility location while still keeping cost (distance in our case) to a minimum.

The program was then run using the objective function, minimize maximum service distance and the same distance option as the previous run and returned the two locations of $(0, 4)$ and $(3, 4)$. These locations enable the emergency vehicle to travel the smallest distance to each emergency location assigned to it. The final run was done using the objective function minimize total cost and returned the locations of $(2, 1)$ and $(3, 5)$. Since cost is not a factor in this problem, this option was calculating the locations based on a sum of overall distance and time. All results can be seen with the original map in Figure 6. It was determined that this objective function along with the maximize demand covered objective worked best for the Rio Rancho problem situation. Since cost is negligible and was entered into the program as a cost of $1 per unit distance, when the program was minimizing total cost, it was actually minimizing total response time based on demand. It is because of this, the objective function for minimizing total cost was used for determining facility locations for the Oak Lawn data.

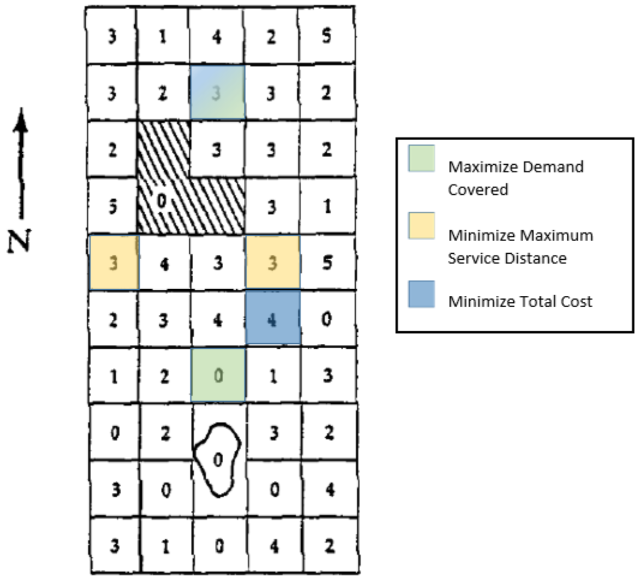


Figure 6: FLP Solver Solutions for Rio Rancho Problem

For the Oak Lawn data the program was run testing a single location for an emergency facility and then ran again for two locations. Once the program was updated to handle the 749 different block locations and the 561,001 different combinations of emergency facilities and location of emergencies, the FLP solution algorithm was executed and the results calculated. To run the program for a data set this large, the CPU time limit needed to be set to a large number allowing the program to run for over 2 days. The program ran a total of 252,210 different iterations to come to the conclusion that block number 425 was suitable for a single emergency facility. From the map in Figure 7, it can be seen that this location is centrally located in the city and makes sense when locating a facility based on arriving at any emergency with the shortest amount of time (fastest option on the FLP solver). It can also be seen how this location is close to the current Police and Fire Station. After many more iterations, the program came to the conclusion that if there were to be two emergency facilities, they should be located at block 487 (39% of the demand) and block 527 (61% of the demand). It can be seen in the Figure 7 these two locations are located practically East and West of each other on opposite ends of the city. The placement of these two facilities also makes sense when you think about how they serve the community. Having locations on either side of town helps minimize response time. They are also located along one of the main streets in the city enabling the responders to quickly exit the facility and navigate towards the intended emergency location on a main artery instead of the slower side streets.

4 Conclusion

Many factors are involved in locating emergency facilities including proximity to community members, zoning, taxes, cost, access to roads, and more. Locating emergency service facilities is a fundamental problem in emergency management. In practice, major disasters (such as fire, earthquake, and flood) often cause enormous property losses. It is the goal of city planners to reduce that loss by locating facilities in appropriate locations. It is important to look at past data to validate or ensure the locations are beneficial to the community for which they serve. When making the decision on where to place an emergency

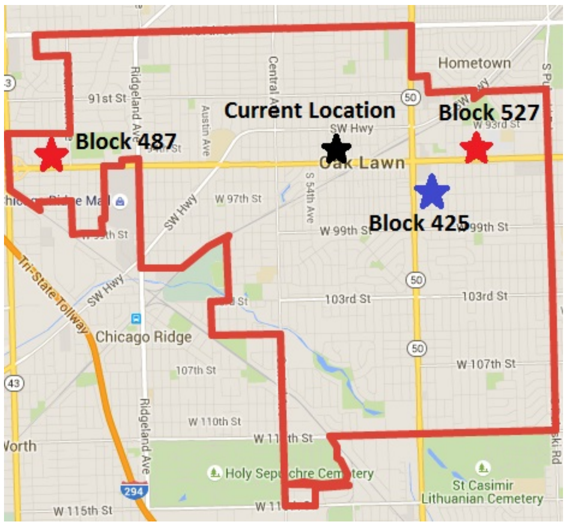


Figure 7: FLP Solutions for Oak Lawn Data. Black Star = Current Location, Blue Star = FLP Solution for One Location, and Red Stars = FLP Solution for Two Locations [13]

facility, many different options are available.

Some useful programs for solving such problems have already been created athough further extension of the FLP solver could prove useful in smaller markets and in research. LoLA is a collection of efficient algorithms for solving planar, network and discrete facility location problems [20]. LoLA can solve a number of different location models including Median, Center, p-median and p-center where p is the number of facilities in a multi-facility problem or the number of objective functions in a multi-objective problem. SITATION is a facility location software that accompanies Daskin's text (Network and discrete location: models, algorithms, and applications) [21]. The SITATION software solves a number of different discrete and network facility location problems including p-median, p-center, set covering, maximal covering, and more. SITATION allows the user to choose from a variety of heuristics and optimization-based approaches for each of the different models. S-distance is a standalone

Spatial Decision Support System, mainly focused on location-allocation analysis [22]. S-distance is able to solve quite large classical discrete and network location-allocation problems, including, p-median, p-center, maximal covering and multi-objective. The current version of the software (version 0.7) offers a number of heuristic and optimization-based algorithms such as greedy and randomized algorithms, local search heuristics, meta-heuristics, and Lagrange relaxation.

Other facility location software is available specializing in solving specific problems and are limited more in their functionalities than the previous three.

- RLP is a program package for solving restricted 1-facility location problems in a user friendly environment [23].
- Optimal locating air polluting facilities is a general modeling system to evaluate and optimize the location of an air polluting facility [24].
- Jure Mihelic's k-center algorithms is a program for solving k-center location problems [25].
- Minimum enclosing circle applet is a program package for solving the Minimal Enclosing Circle problem [26]. It is useful for planning the location of a shared facility.
- Excel template for facility location includes model for center-of-gravity method for locating distribution centers [27].
- GAMBINI is a small GIS-utility which calculates draws and exports multiplicative weighted Voronoi diagrams [28]. A point location data structure can be built on top of the Voronoi diagram in order to find the object that is nearest to a given point.

- Mathematical programming softwares such as CPLEX, LINGO, LINDO and GAMS which are useful when having mathematical models for facility location problems [29] [30] [31].

5 Extension

Due to the large CPU times associated with running this program with more than 200 locations, another algorithm would work faster when dealing with large sets of data. Looking further into large scale p-median problems, an aggregation heuristic could work faster. For large-scale p-median problems, it is common to aggregate the demand points. This size reduction by aggregation makes the problem easier to solve, but introduces error. For the Oak Lawn data, it may make sense to aggregate the demand points so that each node represented multiple blocks in the city. This allows for a solution to have a choice of emergency facility locations which would be helpful in an already largely established community.

For the aggregation problem, three decisions must be made: the number of aggregate demand points, the locations of the aggregate demand points, and the replacement rule [32]. The replacement rule is essentially how you compute the demand for the new aggregate point. This could simply be an addition of all of the previous demands that were aggregated into the new point. It could also include some kind of a business factor or cost. This replacement rule can also deal with how you assign demand to the aggregate point. Error is always in units of the objective function. Typically this makes the most sense for a cost problem where cost can be described in terms of distance. There are no rules on how much error is tolerable and it's usually relative to a previous model or solution. The error can tell you how much the aggregation hurt your result and you can decide if you need to analyze the original problem.

A possible algorithm that could be implemented in the case of the FLP Solver could be a median-row-column (MRC) aggregation algorithm which was presented by France, Lowe and Rayco in 1996 [32]. In their paper they describe the process of how to aggregate demand points to reduce the common error associated with the process. This allows the solver to compute a solution for large data sets. The N-Median Problem, a planar rectilinear version of p-center model is used to seek an aggregation with a small error. The algorithm finds a row-column (rc) median that minimizes the objective function value of the q-median problem with rectilinear distances over all possible rc-medians. MRC is a method for making the three decisions mentioned above.

Appendix A Point Combination Code

```
Sub PointCombinations()

    Dim rInp        As Range
    Dim avInp       As Variant  ' ragged input list
    Dim nCol        As Long     ' # columns in list
    Dim rOut        As Range    ' output range
    Dim iCol        As Long     ' column index
    Dim iRow        As Long     ' row index
    Dim aiCum()     As Long     ' cum count of arrangements
        from right to left
    Dim aiCnt()     As Long     ' count of items in each
        column
    Dim iArr        As Long     ' arrangement number
    Dim avOut       As Variant  ' output buffer

    Application.ScreenUpdating = False

    Set rInp = Range("B4:E13")
    If VarType(rInp.Value) = vbEmpty Then
        MsgBox Prompt:="No input!", _
               Buttons:=vbOKOnly, _
               Title:=sTitle
        Exit Sub
    End If

    Set rInp = rInp.CurrentRegion
    If rInp.Columns.Count < 2 Or rInp.Rows.Count < 2 Then
        MsgBox Prompt:="Must have more than one row and more
             than one columns!", _
               Buttons:=vbOKOnly, _
               Title:=sTitle
        Exit Sub
    End If

    With rInp
        .Style = "Input"
        avInp = .Value
        nCol = .Columns.Count
        Set rOut = .Resize(1).Offset(.Rows.Count + 1)
        Range(rOut.Offset(-1, -1), Cells(Rows.Count,
                Columns.Count)).Clear
    End With

    ReDim aiCum(1 To nCol + 1)
    ReDim aiCnt(1 To nCol)
    aiCum(nCol + 1) = 1
    For iCol = nCol To 1 Step -1
        For iRow = 1 To UBound(avInp, 1)
            If IsEmpty(avInp(iRow, iCol)) Then Exit For
                aiCnt(iCol) = aiCnt(iCol) + 1
        Next iRow

        aiCum(iCol) = aiCnt(iCol) * aiCum(iCol + 1)
    Next iCol

    If aiCum(1) > Rows.Count - rOut.row + 1 Then
        MsgBox Prompt:=Format(aiCum(1), "#,##0") & _
                       " is too many rows!", _
               Buttons:=vbOKOnly, Title:=sTitle
        Exit Sub
    End If

    ReDim avOut(1 To aiCum(1), 1 To nCol)
    For iArr = 1 To aiCum(1)
        For iCol = 1 To nCol
            avOut(iArr, iCol) = avInp((Int((iArr - 1) *
            aiCnt(iCol) / aiCum(iCol))) Mod aiCnt(iCol) + 1,
                 iCol)
        Next iCol
    Next iArr

    With rOut.Resize(aiCum(1), nCol)
        .NumberFormat = "@"
        .Value = avOut
        .Cells(1, 0).Value = 1
        .Cells(2, 0).Value = 2
        .Cells(1, 0).Resize(2).AutoFill .Columns(0)
    End With

    ActiveWindow.FreezePanes = False
    rOut.EntireColumn.AutoFit
    ActiveSheet.UsedRange
    Beep

End Sub
```

Appendix B Total Response Time Code

```
Sub DemandOverResponseTime()
Dim Distance As Double
Dim Demand As Double
Dim TRT As Double
Dim i As Long
```

```
'extract input from the sheet
With Sheets("Combination of Points")
    For i = 15 To 2514

        'select the cell itself
        Distance = Range("G" & i).Value
        Demand = Range("I" & i).Value

        'forumla for total response time is (Distance *
            Demand)
        TRT = (Distance * Demand)

        Range("K" & i).Value = TRT
    Next i
End With

Beep

End Sub
```

Appendix C   Sum of Demand per Block Code

```
Sub SumIfs()
Dim i As Integer, j As Integer

With Sheets("Combination of Points")
    For i = 0 To 4
        For j = 0 To 9
            .Range("M15").Offset(j, i).Value = _
                WorksheetFunction.SumIfs(.Range("K15:K2514")
                   , _
                                         .Range("B15:B2514")
                                             , i, _
                                         .Range("C15:C2514")
                                             , j)
        Next 'j
    Next 'i
End With

With Range("S15")
    .FormulaArray = "=SMALL(IF($M$15:$Q$24<>0,$M$15:$Q$24),
        ROWS($A$1:$A1))"
    .Copy .Offset(1, 0).Resize(2, 1)
End With

Beep

End Sub
```

References

[1] L. Caccetta and M. Dzator, "Heuristic methods for locating emergency facilities," in *Proceedings Modsim*, vol. 2005, Citeseer, 2005.

[2] NFPA, *NFPA 1710 Standard for the Organization and Deployment of Fire Suppression Operations, Emergency Medical Operations, and Special Operations to the Public by Career Fire Departments.* Quincy, 2010 ed., 2010.

[3] M. S. Daskin and K. L. Maass, "The p-median problem," in *Location Science*, pp. 21–45, Springer, 2015.

[4] S. L. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph," *Operations research*, vol. 12, no. 3, pp. 450–459, 1964.

[5] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. i: The p-centers," *SIAM Journal on Applied Mathematics*, vol. 37, no. 3, pp. 513–538, 1979.

[6] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. ii: The p-medians," *SIAM Journal on Applied Mathematics*, vol. 37, no. 3, pp. 539–560, 1979.

[7] C. Hosage and M. Goodchild, "Discrete space location-allocation solutions from genetic algorithms," *Annals of Operations Research*, vol. 6, no. 2, pp. 35–46, 1986.

[8] B. L. Golden and C. C. Skiscim, "Using simulated annealing to solve routing and location problems," *Naval Research Logistics Quarterly*, vol. 33, no. 2, pp. 261–279, 1986.

[9] F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

[10] F. Glover, "Tabu search: Part ii," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4–32, 1990.

[11] E.-G. Talbi, *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

[12] B. A. Fusaro, "The 1986 mathematical competition in modeling," *Mathematical Modeling*, vol. 7, no. 4, pp. 537–543, 1986.

[13] Google, "Map of oak lawn, illinois," 2015. https://www.google.com/maps/place/Oak+Lawn,+IL/@41.7090825,-87.759552,13z/data=!4m2!3m1!1s0x880e315112a50b61:0x1e773c4106829929.

[14] F. S. Hillier. and G. J. Lieberman, *Introduction to operations research.* New York: McGraw-Hill, eighth ed., 2005.

[15] C. R. Reeves, “Improving the efficiency of tabu search for machine sequencing problems,” *Journal of the Operational Research Society*, pp. 375–382, 1993.

[16] G. L. Michel Gendreau and R. Séguin, “A tabu search heuristic for the vehicle routing problem with stochastic demands and customers,” *Operations Research*, vol. 44, no. 3, pp. 469–477, 1996.

[17] M. Sun, “Solving the uncapacitated facility location problem using tabu search,” *Computers and Operations Research*, vol. 33, no. 9, pp. 2563–2589, 2006.

[18] A. Hertz, E. Taillard, and D. D. Werra, “A tutorial on tabu search,” in *Proc. of Giornate di Lavoro AIRO*, vol. 95, pp. 13–24, 1995.

[19] G. Erdogan, “Flp spreadsheet solver,” 2014.

[20] H. W. Hamacher, K. Klamroth, S. Nickel, and A. Schoebel, “Library of location algorithms,” tech. rep., University of Kaiserslautern, 1996. http://www.mathematik.unikl.de/ wwwwi/WWWWI/DFG/lola.html.

[21] M. S. Daskin, “Sitation-facility location software. department of industrial engineer-ing and management sciences.” Northwestern University, Evanston, IL, 2002. http://users.iems.northwestern.edu/msdaskin/.

[22] P. Y. N. Sirigos S., “S-distance software.” Department of Planning and Regional Development (DPRD), University of Thessaly, Greece, 2005.

[23] S. Nickel and H. W. Hamacher, “Rlp: a program package for solving restricted 1-facility location problems in a user friendly environment,” *European Journal of Operational Research*, vol. 62, no. 1, pp. 116–117, 1992.

[24] J. Fliege, “Olaf–a general modeling system to evaluate and optimize the location of an air polluting facility,” *OR-Spektrum*, vol. 23, no. 1, pp. 117–136, 2001.

[25] J. Mihelic, “Jure mihelic k-center algorithms.” Department of Computer and Information Sci-ence, University of Ljubljana, 2004.

[26] J. Eliosoff and R. Unger, “Mec ? minimum enclosing circle applet,” 1998. http://www.cs.mcgill.ca/cs507/projects/1998/jacob/welcome.html.

[27] Microsoft, “Microsoft excel,” 2010. Redmond, Washington.

[28] B. N. Boots and M. Tiefelsdorf, “Gambini-programme for calculating multiplicative weighted voronoi diagrams,” 1997. http://www.wlu.ca/wwwgeog/special/download/gambini.htm.

[29] IBM, “IBM CPLEX optimizer,” 2015.

[30] L. Systems, “LINGO/LINDO,” 2015. Chicago, Illinois.

[31] GAMS Development Corporation, “GAMS,” 2015. Washington DC.

[32] T. J. L. Richard L. Francis and M. B. Rayco, “Row-column aggregation for rectilinear distance p-median problems,” *Transportation Science*, vol. 30, no. 2, pp. 160–174, 1996.