Mini project 2: Go Greedy Applying the Greedy Approach to Real-World Problems

(Option 2)

Data Structures and Algorithms | COP3530

Austin Rumancik | Spencer G Sherman

Presentation on December 14, 2022

The number of participants: 2 students per project; the grade will be assigned at an individual level depending on the degree of contribution.

Presentation deadline: Same as the Final Exam

Deliveries

- A program in C++, MATLAB, or Python.
- The program must visualize the coverage that results from using your smart data acquisition mechanism.
- An experiment shows that your approach outperforms the area or sensor coverage of the naïve (pure greedy, and random) data acquisition policies while optimizing the budget.
- Presentation of your approach.
-

Introduction

This mini-project explores the solution to a real problem using the programming tools and paradigms you have learned so far.

Rubric

This rubric only applies after the presentation; no presentation will result in a zero in this project.

- This project counts as FINAL EXAM and Mini Project 2.
- Program running: 50%
- Algorithm design: 20%
- Experiment design: 20
- Report: and presentation 10%

Learning Outcomes

Upon completing this mini-programming project, you will be able to:

1. Apply algorithm design to tackle a real problem.
2. Implement a solution and choose the programming language that better fits the problem requirements.
3. Apply a greedy or Dynamic programming approach to solve a problem.
4. Experiment with design to demonstrate that your solution outperforms a given approach

Problem Description

Orlando is becoming a smart city, namely, a city whose services and data support decisions. Thus, the town is interested in acquiring sensor readings of pollution at high granularity.

One of its main goals is to get a more accurate sense of the pollution levels in the city. Thus, the town subcontracts a data collection campaign to a new startup company in Lakeland. The company's name is (Data Structures & Algorithms -FALL2022). They specialize in profitable high-tech projects with a focus on algorithm design.

Sketching a data collection campaign

The first decision is to collect data with high granularity and accuracy. Thus, the company decides to crowdsource the data collection to all the citizens of Lakeland who own a smartphone. The recruitment campaign follows the next steps.

1. The company released an app that interested participants can download and install on their smartphones.
2. The app enables the necessary phone sensors to collect pollution data (assumption: all the phones have a sensor for that purpose)
3. The app allows participants to send to the company the following information: the sensing measurements, its location, and the bid price (asking price for the collected sample)
4. The company checks the data quality, selects the winners based on their competitive bid price, and notifies the selected winners.
5. The selected winners receive their bid price (payment for the samples).

The following graph shows an example of the locations of participants in a given city. Every participant submits the sensing sample location (GPS) and the collected sample's bid price.

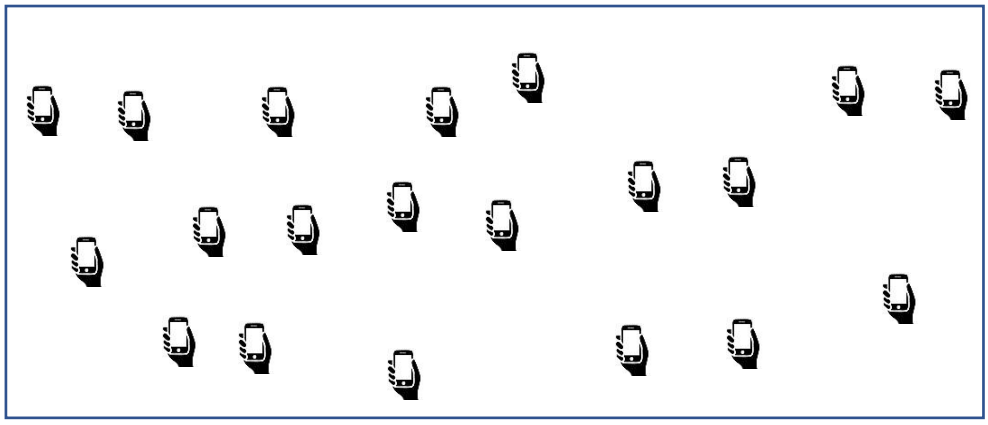


Figure1. Case 1 Sketch of participants whose locations follow a uniform distribution.

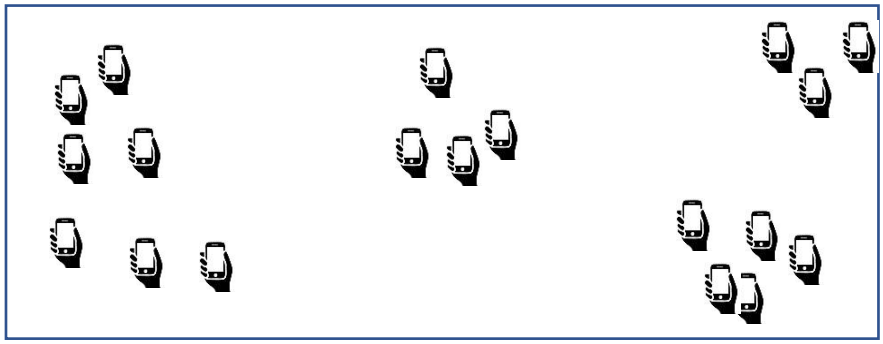


Figure 2. Case 2. More realistic distribution, people usually are in neighborhoods more clustered

Sensor model

We assume each phone sensor has a radius R at which is the sensing confidence

$$f(d(u_i, u_j)) = \begin{cases} 1 & \text{If } d(u_i, u_j) \leq R \\ 0 & \text{Otherwise} \end{cases}$$

Where $d(u_i, u_j)$ is the Euclidean distance between a sensor u_i and another sensor u_j , and $R > 0$ is a constant that defines each sensor's coverage area. Indeed, this function establishes a disk centered at sensor ui with radius R. All sensors within such a disk have a coverage measure of 1 and are said to be covered by sensor ui. On the other hand, all sensors outside of such a disk have a coverage measure of 0 and are said to be not covered by this sensor.

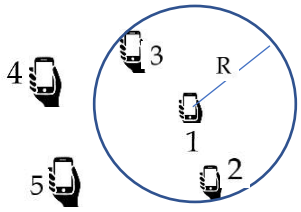


Figure 3. Sensor Model

Here, Figure 3 shows how the sensor centered at 1 covers sensors 1, 2, and 3, while sensors 4, and 5 are not covered.

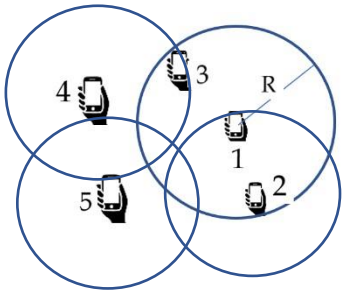


Figure 4. Sensor coverage when taking into account the disk of every sensor.

Figure 4 shows that sensor 4 covers sensor 4 and sensor 3, sensor 5 covers itself, and sensor 2 covers sensor 2 and sensor 1. Finally, sensor 1 covers sensor 1, sensor 2, and sensor 3. Thus, to cover all the sensors, we will have to buy the data of sensors 1, 2, 3, 4, and 5; however, we don't have the budget to buy all these samples. The question here is how to optimize the budget while maximizing the number of sensors covered.

Note that the sensing data provided by any phones within the area coverage (with the disk) of the center's phone is very similar.

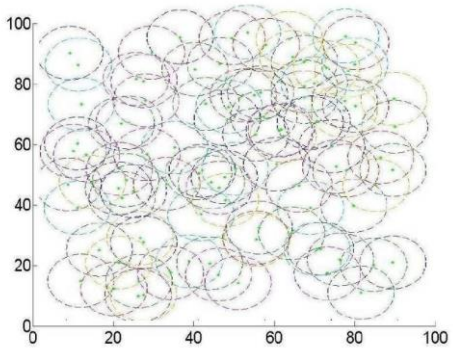
Data acquisition Model

Here, the company allocates a budget and wants to acquire a representative set of samples within the given budget. However, if you wish to have a representative group of samples, you need to get samples from the different locations of the city, in order words, you want to maximize the area coverage.

Assumptions about the income distribution of the city

Let's assume that we have different neighborhoods with different average income distributions.

Thus, participants in low-income neighborhoods will ask for a low price for sensing samples, while those living in high-income areas will ask for a high price.



Naïve acquisition Model 1

We are acquiring all the samples offered by participants. Here, we have perfect coverage (see figure). However, we don't have the money to buy all the samples, we have a limited budget, and we want to have some profit.

Naïve approach Model 2: Slopy greedy

Use the budget to buy the cheapest sensing samples in ascending order until exhausting the budget. Figure 6 shows the result of this acquisition policy, given the city's income distribution.

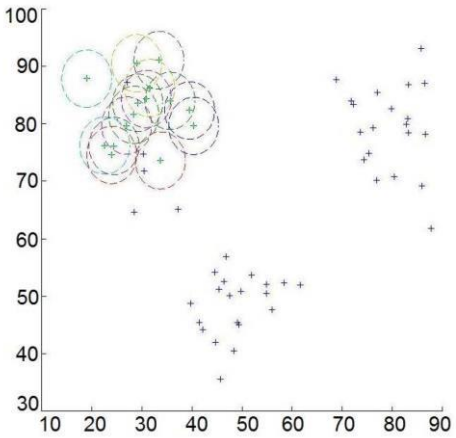


Figure 6. Data acquisition policy: Pure greedy

Here, we acquired a lot of sensing samples because participants from a populated low-income neighborhood were asking too low for their samples. Unfortunately, these readings do not constitute a representative sample of the city's pollution. We don't have samples of many places.

Your job (find a sub-set of samples that maximize coverage at minimum cost)

Design an algorithm to acquire sensing samples that maximize coverage (area or sensors) while minimizing the cost of acquisition. The running time of an exhaustive search (try all the combinations) is exponential and, therefore, not a feasible solution.

So, the company's job is to use engineering talent to design a solution (maybe greedy) to solve the problem.

Develop a solution, and show that your solution is better than

1) a random acquisition policy

2) better than a pure slopy greedy policy.

To show that your solution is better, you should generate the participants using a uniform and multinormal distribution (generate the locations and the bits prices in clusters) to resemble the location of participants in low- and high-income neighborhoods.

Suggested strategies

This problem (find a sub-set of samples that maximize coverage at minimum cost) is a combinatorial problem. So, the exhaustive search for an optimal solution is exponential, as we learned during the first class of dynamic programming when reviewing the knapsack problem. Thus, a greedy solution getting us a sub-optimal (approximation) solution is acceptable. Therefore, you can try a greedy or a dynamic programming solution.

Check if any of these approaches could be helpful for you.

- Knapsack
- Set cover
- Budgeted Max

Tasks: Simulation

Generate the (GPS) locations (uniform, distributed, and clustered). Everything happens in a 100x100 grid

Generate the bid prices (cluster (normal distribution per cluster, and uniform distribution))

Other decisions to take

Budget=

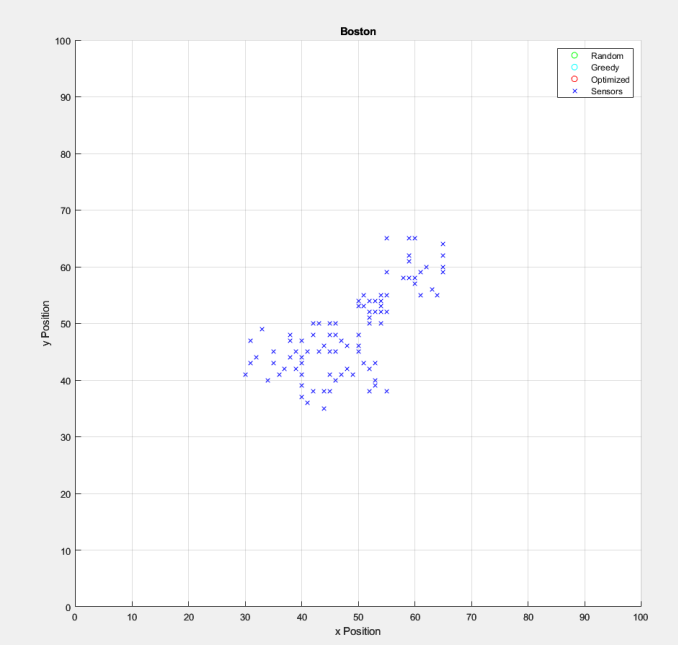
Sensor Radius=

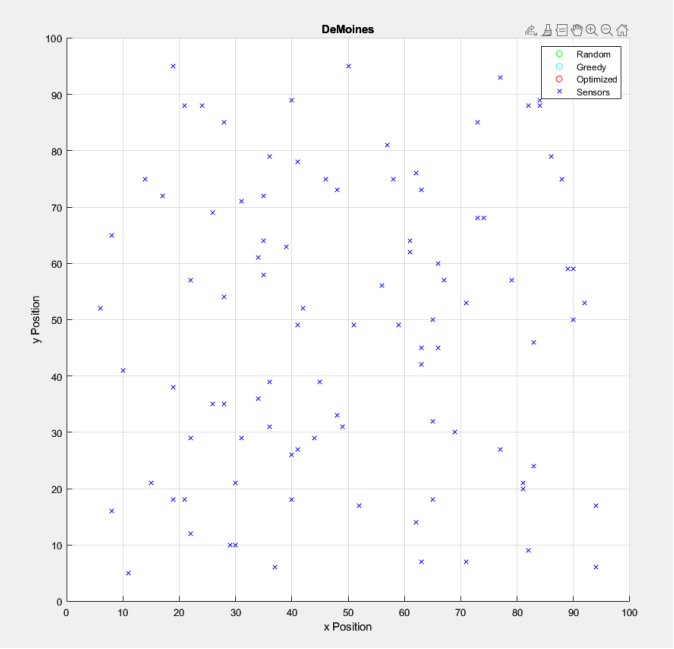
What are you covering: area? Or maybe other participants

What to do if you have some remaining money after buying the sensing samples.

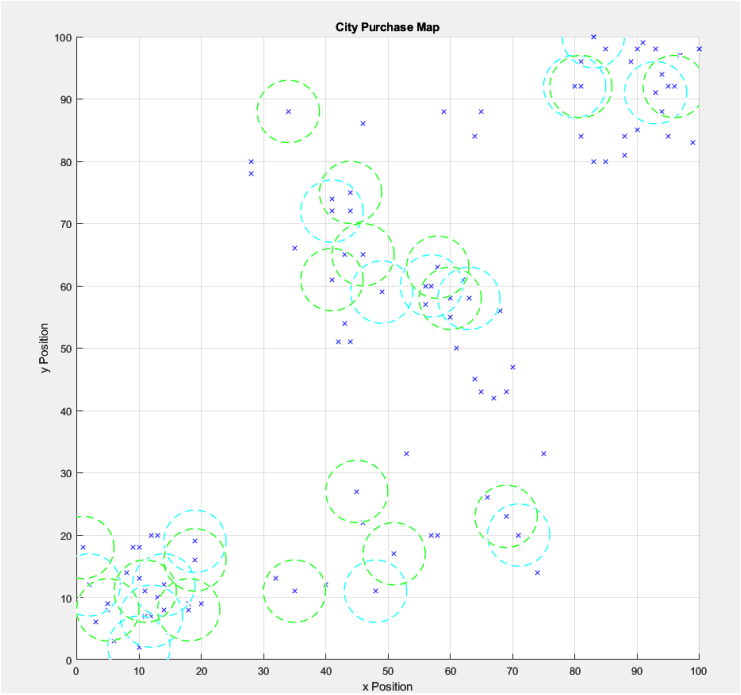
Student Approach

Given what we have observed/learned through course lectures and given the set parameters around the project outlined above, we have taken specific steps to optimize the coverage potential of our data model. There are many things regarding even just the planning behind building out the cities. We took an approach in which we could modularize the creation of each map providing us the chance to see a multitude of different sensor distributions and how each selection approach would perform. To do this, our constructor outlining the creation of each sensor (which are all classes themselves and all contained within a vector set representative of a city) has user defined traits for locality, bid price, distance to other nodes, etc (randomized within a set range). We then moved on with gathering physical data for the topology of various cities and inputting them into our solution to garner actual testable values (see two examples of the sensor distribution below).

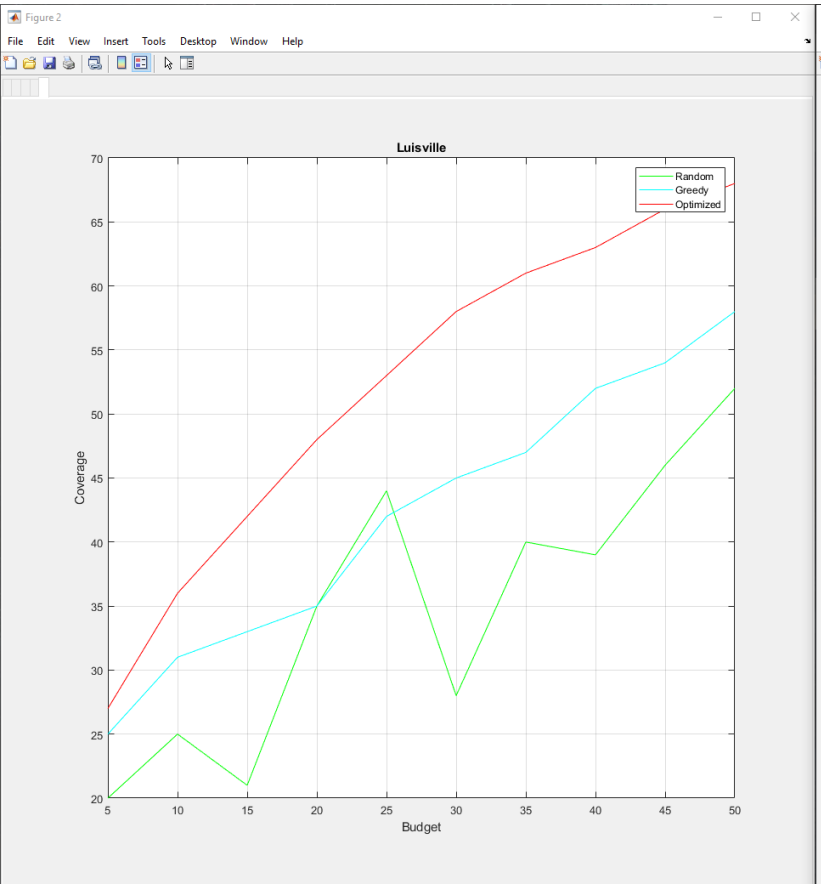


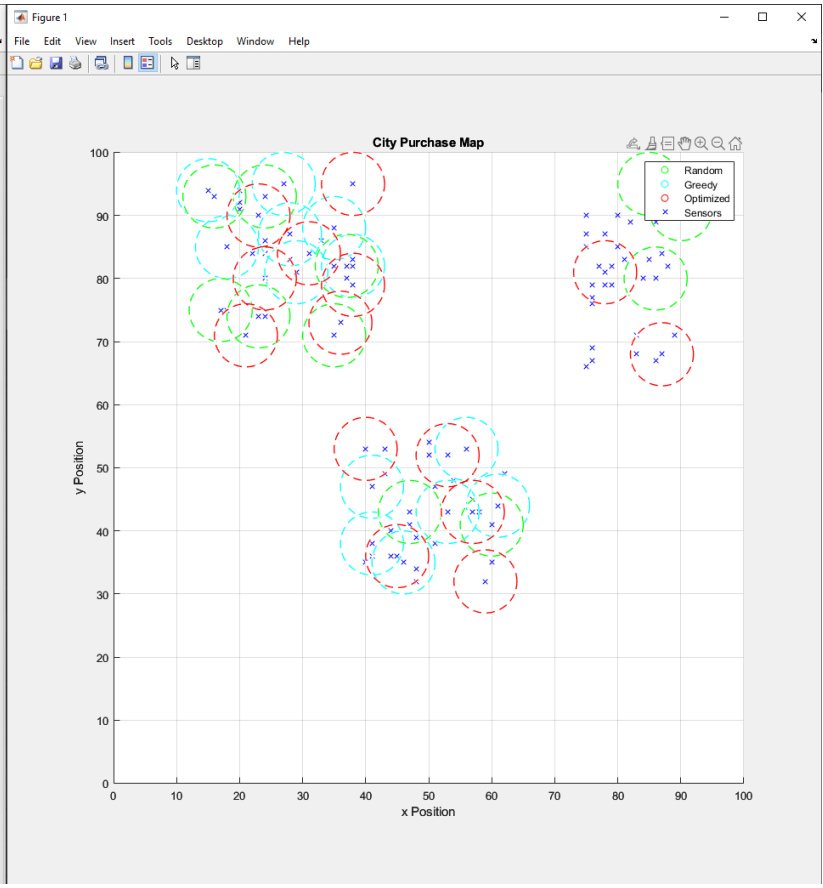


Next, we moved on to conquer the actual algorithm to be implemented. With the task at hand, we focused more on optimizing the resulting coverage for allotted budget and less on the computational weight of each process. Thus, although the complexity of our algorithm leaves some simplicity to be desired, the results are apparent in the pure value of each purchase. The two naïve approaches provided were based on randomization and pure greedy. For these, we created two function calls which take a defined city and allotted budget to generate purchase selections. The randomized approach simply grabs any randomly indexed sensor within the containing city and purchases it until there is no more money to be spent. The greedy implementation sorts the city that was passed in by requested bid price and makes its selections through a minimum favored logic. It should be noted that we instilled various limitations such as the fact that a covered node cannot be purchased or counted twice to stay consistent with a realistic representation (the purchases can be seen below as a basic example).



For our optimized implementation, we took inspiration from both set coverage and the famous knapsack problem. We kept a running collection of our selections as well as a dynamically updated weighted city sort that will keep our most effective potential purchase as the next sensor to be bought (sensor coverage per cost). We were initially going to error check these with unique key and value pairs, but we found that this was not needed for an application of this scale. We will sometimes have a little remaining money (but we resolved this by spending on whatever we can before we have no purchasable nodes), but the results below clearly represent the performance of each algorithm. For reference, each sensor has a radius of 5, our variable asking prices are spatially dependent (neighborhoods), and we are looping through potential budgets ($5.00-$50.00) to gather the following data. All sensor positions and purchase decisions are exported and formatted through Matlab to visualize each distribution.





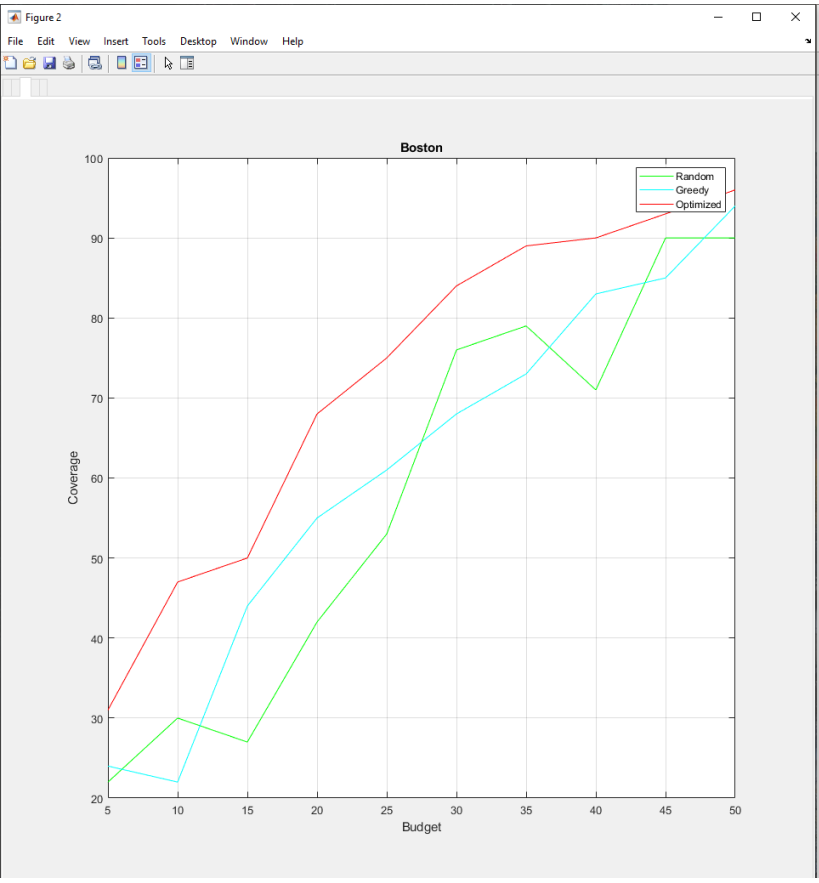


Figure 2
File Edit View Insert Tools Desktop Window Help
Boston
Random
Greedy
Optimized
Coverage
Budget

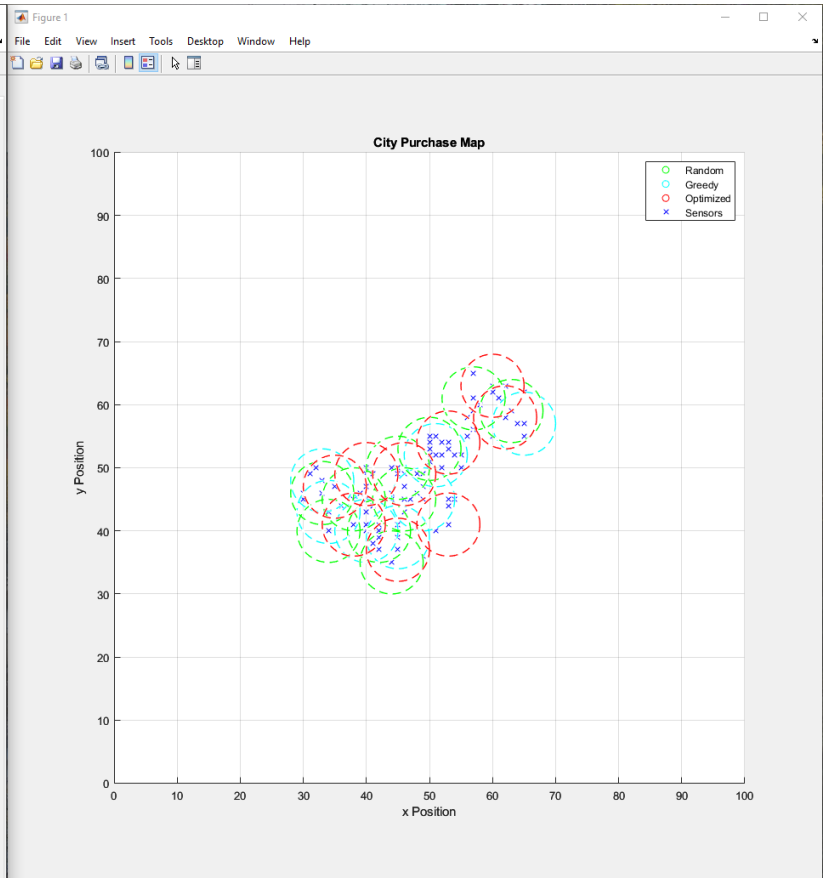


Figure 1
File Edit View Insert Tools Desktop Window Help
City Purchase Map
Random
Greedy
Optimized
Sensors
y Position
x Position

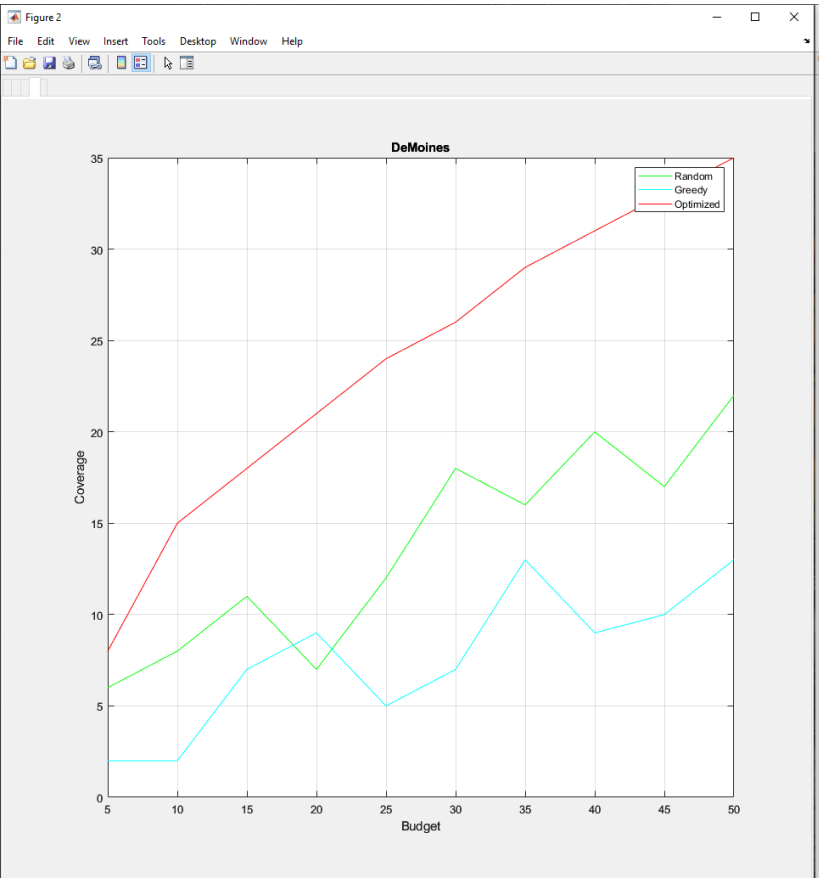


Figure 2
File Edit View Insert Tools Desktop Window Help
DeMoines
Random
Greedy
Optimized
Coverage
Budget

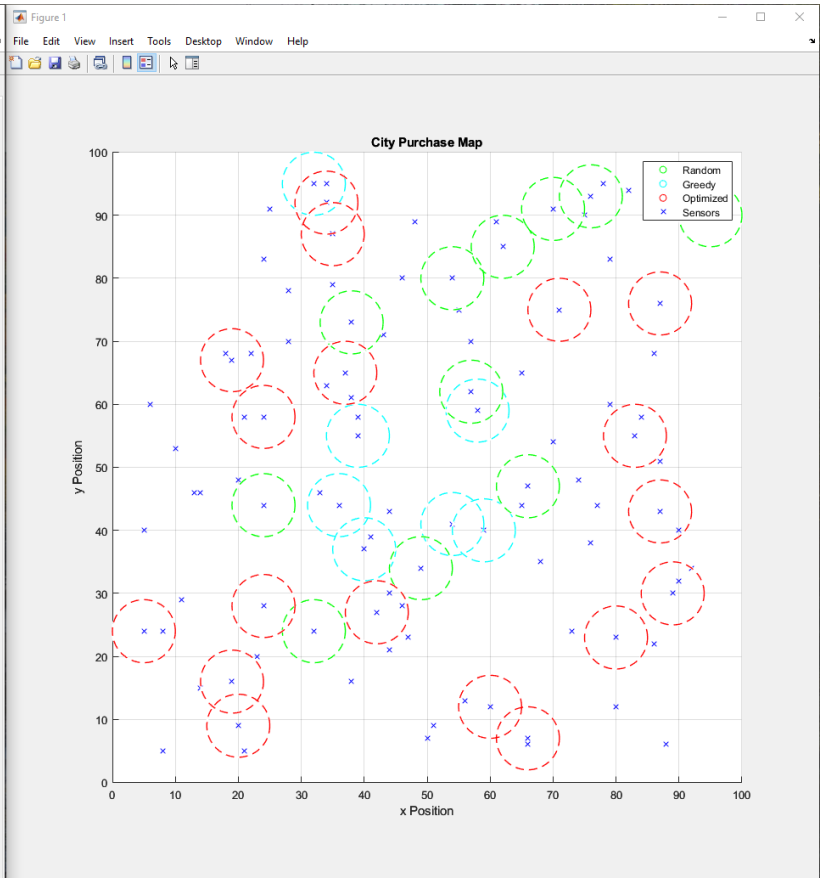


Figure 1
File Edit View Insert Tools Desktop Window Help
City Purchase Map
Random
Greedy
Optimized
Sensors
y Position
x Position

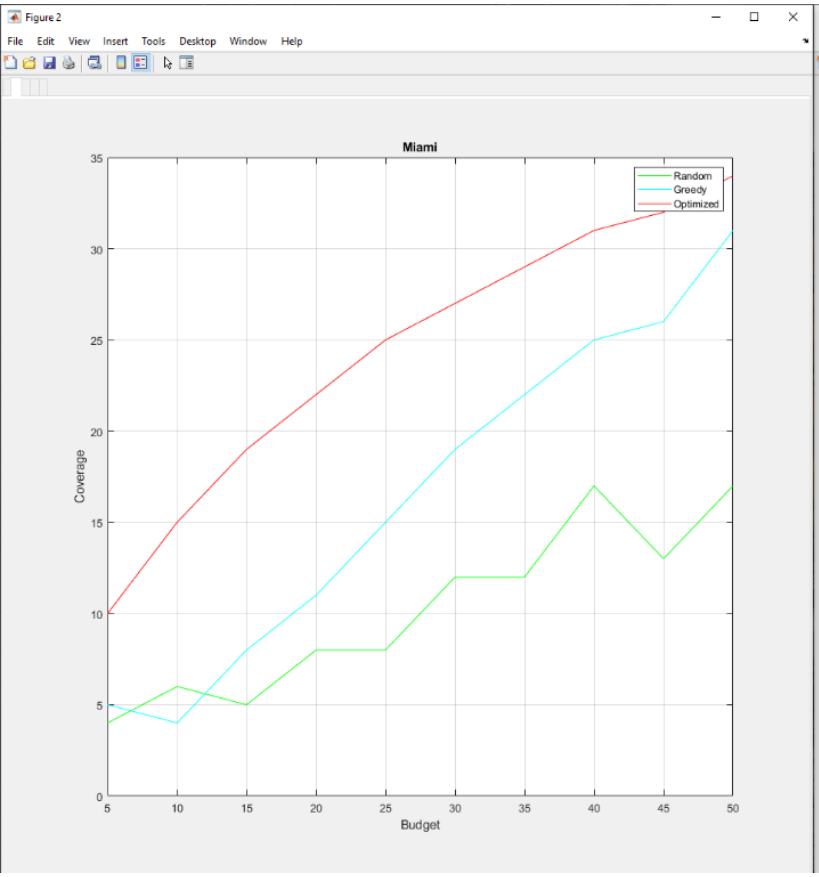


Figure 2
File Edit View Insert Tools Desktop Window Help
Miami
Random
Greedy
Optimized
Coverage
Budget
0 5 10 15 20 25 30 35
5 10 15 20 25 30 35 40 45 50

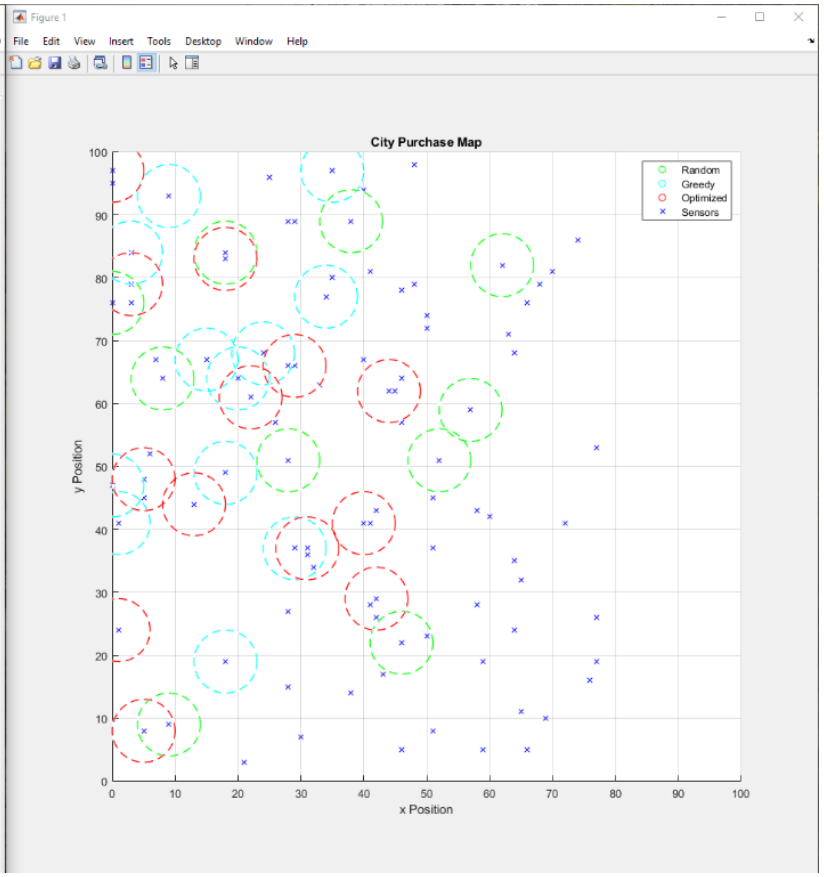


Figure 1
File Edit View Insert Tools Desktop Window Help
City Purchase Map
Random
Greedy
Optimized
Sensors
y Position
x Position
0 10 20 30 40 50 60 70 80 90 100

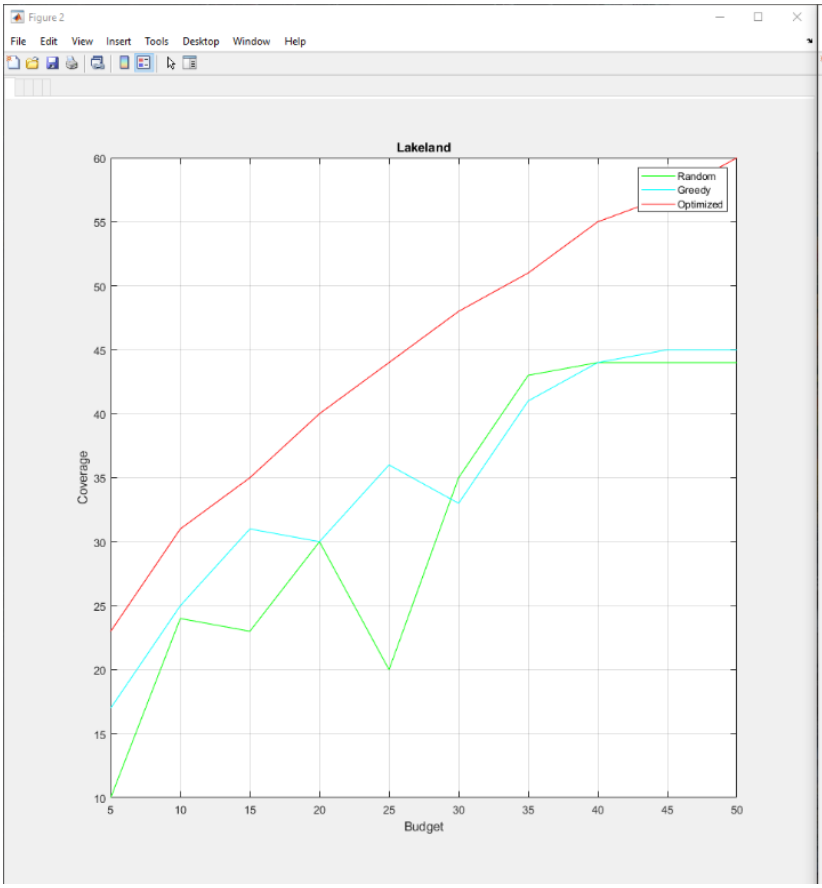


Figure 2
File Edit View Insert Tools Desktop Window Help
Lakeland
Random
Greedy
Optimized
Coverage
Budget
10 15 20 25 30 35 40 45 50 55 60
5 10 15 20 25 30 35 40 45 50

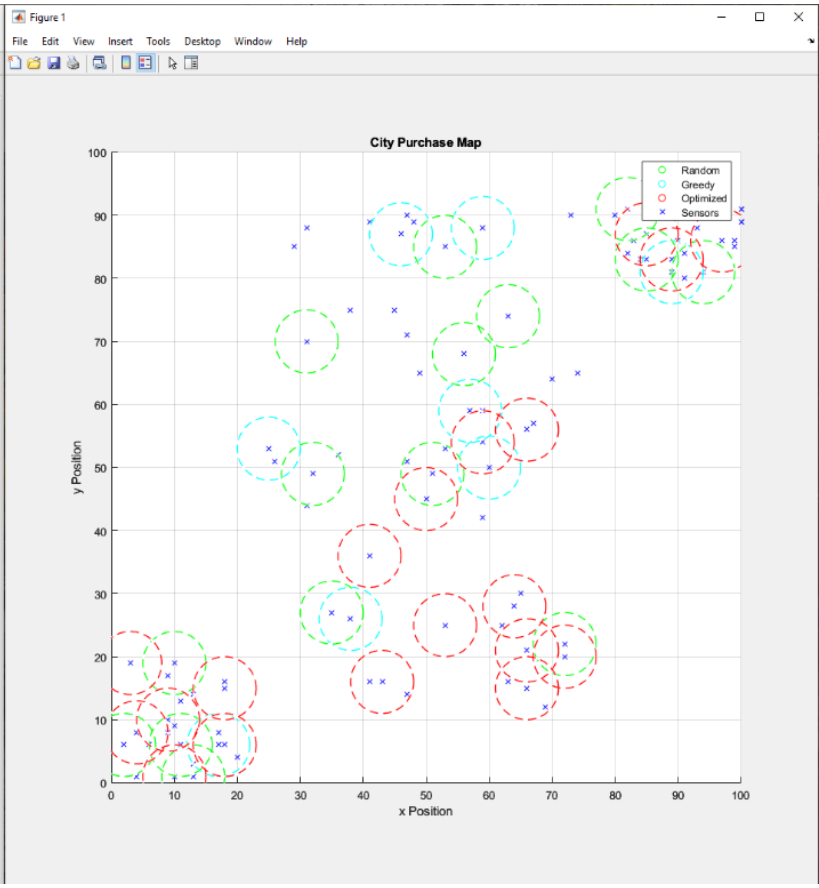


Figure 1
File Edit View Insert Tools Desktop Window Help
City Purchase Map
Random
Greedy
Optimized
Sensors
y Position
x Position
0 10 20 30 40 50 60 70 80 90 100