Lagrangian Relaxation Implementation

Team members: Ghizlaine Bennani, Chuiyi (Tracy) Liu, Tate Campbell & Abhishek Singh

Abstract

The Maximal Covering problem is one of the very early location covering models studied in the vast literature on models that provide optimal coverage to several demand areas. A demand is said to be covered in a specific area if it is within a required distance from the facility and is connected to the facility. This project aims to design a Lagrangian relaxation heuristic for the case of Maximal Coverage problem so as to maximize the demand met. To do so, we need to locate a specific number of plants in a network of nodes that would maximize the number of demand nodes that are serviced within a fixed standard distance or time period. The Maximal Covering problem is used in applications such as the optimal location of emergency response facilities, services, and vehicles; retail stores; communication networks; biology conservation, security monitoring sensors etc. For implementation purposes, we have been working with 3 differents scenarios [49 Nodes, 88 Nodes, 150 Nodes] and we have the demand & distance matrix. In this context, a relevant objective function would be the maximization of demand met under the constraints of finite tour length. The demand we could meet are [136 million, 22 million, 23 million] for the 3 scenarios respectively using our algorithm.

Introduction

The Maximal Covering problem is typically a hard or computationally expensive problem for large datasets with multiple demand & supply nodes. For the problem at hand, we have information on the demand nodes, number of supply points & the distance between the demand nodes. We want to ensure that we cover the maximal possible demand travelling lesser distance with the limited number of supply points. So our solution shall incorporate the above requirements for the 3 scenarios at hand:

1. 49 Nodes
2. 88 Nodes
3. 150 Nodes

However, we need to design a solution that best navigates this issue and produces reliable results in limited amount of time we have. Also, approximation algorithm like Lagrangian Relaxation benefit from the initial set of parameters chosen or assumed. Hence the solutions we produce could be considered local optimal solutions and may not be global optimal solutions owing to the convergence criterion that we set, that guarantee performance for many scenarios. Major applications of the problem we are solving would be warehouse & retail outlet network for large corporations that are shipping their ready made goods from warehouses or production points to retail outlets.

Model Description

Lagrangian Relaxation is a relaxation algorithm that is applied to this complex problem of demand maximization. This is an iterative algorithm like Gradient descent where in each iteration it tries to improve its cost function obeying the problem constraints. For this problem with (49, 88, 150) nodes with their demand values we strive to maximize the demand met by converging to the best network of supply nodes that span the maximum demand nodes. We conclude by reaching a good enough solution within limited time & computational power. It is however unlikely to reach global optima with this algorithm. Initial assumptions (Cutoff distance, K (Number of iterations for upper bound), alpha value drive algorithm performance.

Pseudo Code

We have programmatically used the below steps to converge to our solution. Symbols: a_{ij} : distance matrix a (i row, j column), h_i: demand at node i, λ_i: Lagrange Multiplier

Step 0: Load the distance & demand matrix (Only 1st column) and preprocess the distance matrix to turn into a sparse matrix of 1's & 0's.

Step 1: a_{ij}[Dist_a_{ij} > 700] -> 0, else 1 ;

Step 2: For each Demand Node a_i, compute $\lambda_i = h_{mean} + .5(h_i - h_{mean})$

Step 3: For each Demand Node a_i, compute Z_i: if $(h_i - \lambda_i) > 0$ then 1 else 0

Step 4: For each Demand Node a_i, compute $(h_i - \lambda_i) * Z_i$ leading to $\Sigma[(h_i - \lambda_i) * Z_i)]$ (Our 1st part of Objective function, which is to be maximized)

Step 5: For each Demand Node a_i, compute $\Sigma a_{ij.} \lambda_i$ (dot product of distance matrix column **i** and λ_i)

Step 6: For each Demand Node a_i, Assign X_j to (1,0) with constraint $\Sigma X_j <= 10$ (chosen P)
and with the objective to maximize $\Sigma((\Sigma a_{ij.} \lambda_i)X_j)$ (2nd part of Objective function)

Step 7: For each Demand Node a_i, compute $\Sigma a_{ij.} X_j$ (dot product of distance matrix column **i** and X_j)

Step 8: For each Demand Node a_i, compute $(\Sigma a_{ij.} X_j - Z_i)$ (Previous step - Z_i)

Step 9: Compute Best Upper Bound (BUB) by adding results from **Step 4** & **Step 6**

Step 10: Compute Best Lower Bound (BLB) by adding Sum of demands met by current supply Nodes (h_i)

Step 11: Initiate $\alpha = 2$, to begin your algorithm and reduce its value to half, if (BUB) doesn't reduce substantially after 4 iterations

Step 12: Compute $t^* = (\alpha * (BUB - BLB)) / (\Sigma (\Sigma a_{ij.} X_j - Z_i)^2)$

Step 13: Compute the difference between BUB & BLB call it Δ

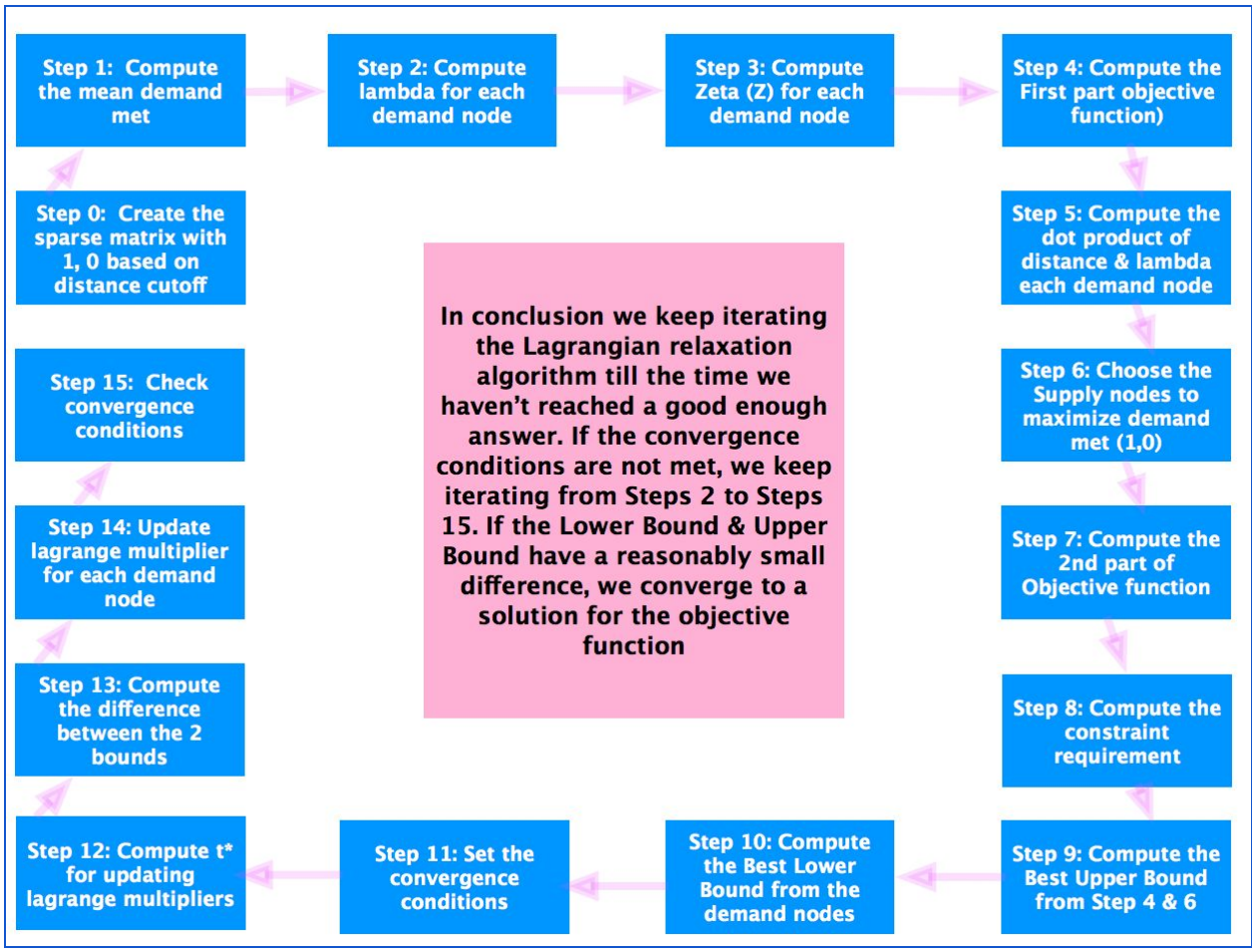
Step 14: For each Demand Node a_i Compute λ^*_i using $[\lambda^*_i = \max(0, \lambda_i - t^*(\Sigma a_{ij.} X_j - Z_i))]$

CStep 15: If Δ is too small or Maximum number of iterations has been exceeded, stop the process & report the solution, else repeat **steps 2 - 15**

Solution Algorithm

Our Solution algorithm was programmatically generated in IPython using basic python libraries and the bulk of the algorithm was prototyped from scratch by us. We first started by loading & cleaning the datasets concerned and then wrote our own functions to implement Lagrangian Relaxation for this Problem. Schema 1 walks through our process to arrive at the solution.

Schema 1. Diagrammatic walkthrough of Lagrangian Relaxation



Key points from this algorithm would be:

1. Computation time & memory utilization increase with higher Node size
2. Initial assumptions help us arrive at the final solutions (local optima)

Please feel welcome to refer to our code **Lagrangian_Relaxation.py,** to programmatically evaluate our solution. To run the code:

1. Delete the header row from all the demand data files
2. Place the code & data in the same directory and run the code: (Node_type = 49, 88, 150)

```
$ python Lagrangian_Relaxation.py 49 700 10 #Node_type Distance_Cutoff Supply_points
```

Computational Results / Findings

Project Results

We could successfully solve this problem of maximal covering for each of the 3 scenarios of 49 Nodes, 88 Nodes, 150 Nodes. What we managed to do here is that we maximized the demand covered in the network from 10 supply nodes (default) within our distance constraints (default). Table 1 summarizes of the findings from this analysis.

Table 1. Numeric summarization of lagrangian relaxation performed on 49, 88 and 150 node networks.

Scenario	Final Upper Bound	Final Lower Bound	Iterations Performed	Total Demand	Time Taken	Nodes Covered
49 Nodes	135,489,556	135,488,394	500	247,051,601	8 secs	26
88 Nodes	21,986,816	21,872,554	500	44,840,571	10 secs	28
150 Nodes	23,251,318	21,892,554	500	58,196,530	12 secs	27

Network Visualization

Graphs for the 49 node and 88 node datasets were visualized in Python using the networkx graphing library. Weights between nodes were computed using the ratio of distances to the total sum of squared distance values. Weight thresholds for drawing edges were chosen as 0.030 for the 49 node graph and 0.025 for the 88 node graph, the output of which are shown in Figures 1 and 2, respectively.

Figure 1. 49 node network graph using a weight threshold of 0.030.

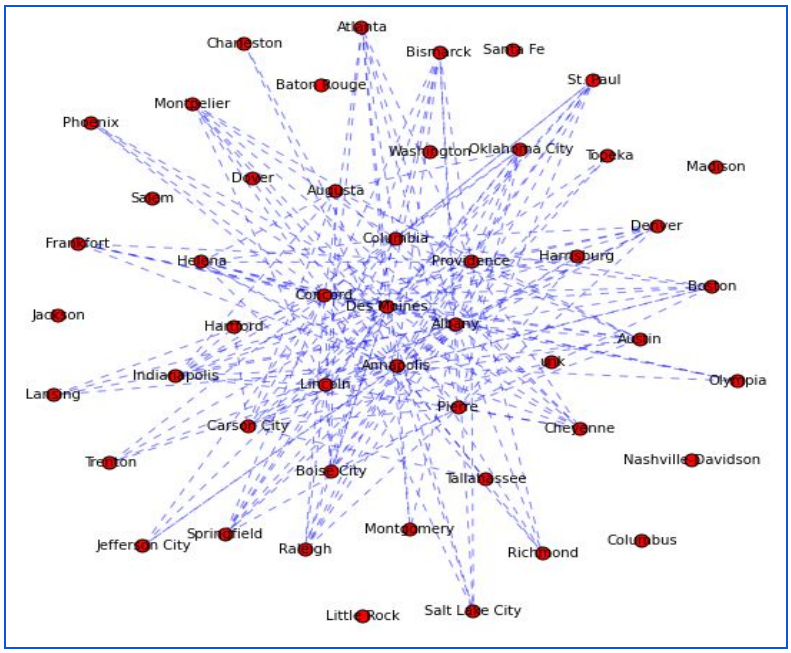
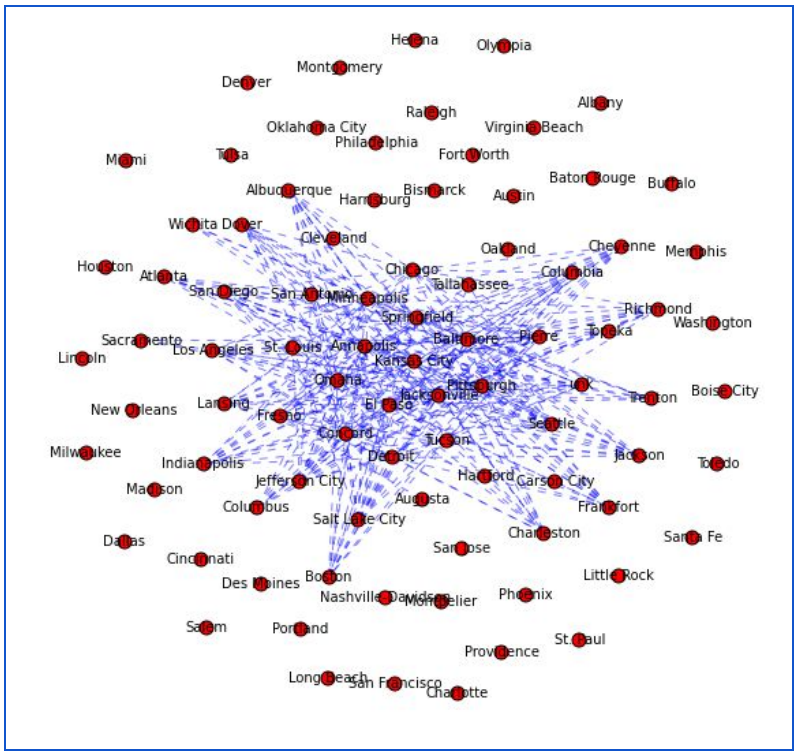


Figure 2. 88 node network graph using a weight threshold of 0.025.



Sensitivity Analysis

Several parameters would directly influence the output of our algorithm. To get some insights of this influence, we have performed a sensitivity analysis respectively on the 49 nodes network and the 88 nodes network. For both networks, we have first fixed the distance cutoff to 700 and measured the total demand met while the number of visited nodes varies from 5 to 45 with an increment of 5 nodes at each step. The results are presented in Figures 3.a and 3.b. Next, we have fixed our number of visited nodes to 10 and varied the distance cutoff from 100 to 2000 with an increment of 200 at each stage. The results are presented in Figures 4.a and 4.b.

Figure 3.a. Demand in millions vs. distance threshold plot for 49 node network.

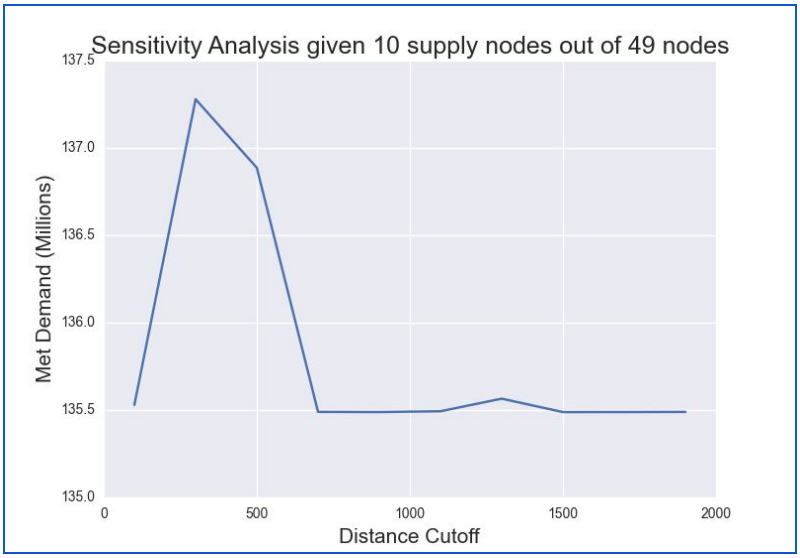


Figure 3.b. Demand in millions vs. number of nodes plot for 49 node network.

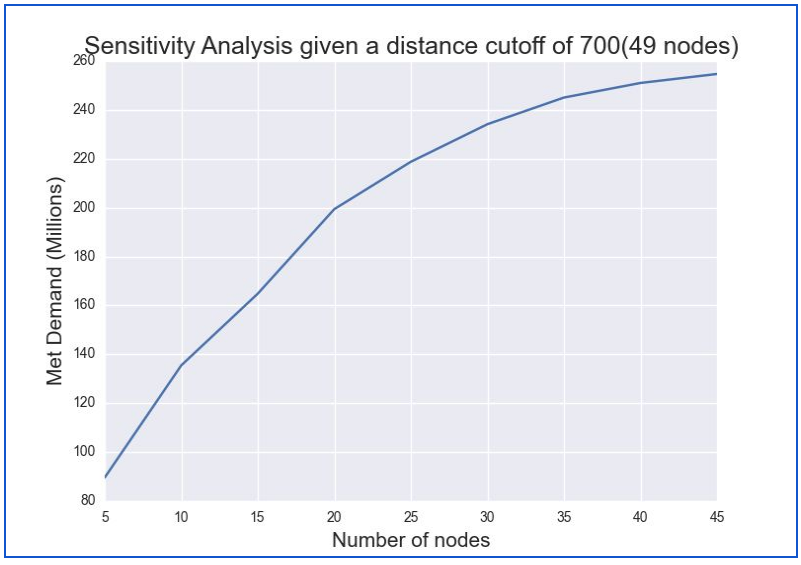


Figure 4.a. Demand in millions vs. distance threshold plot for 88 node network.

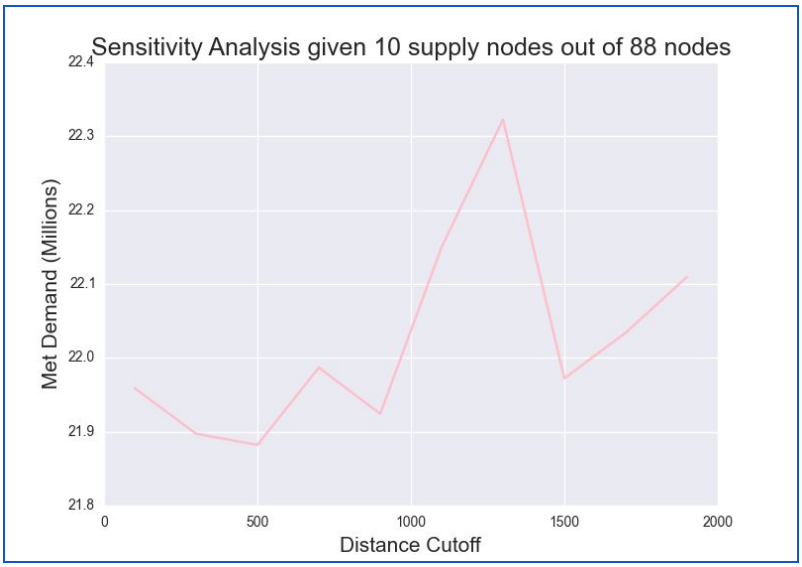
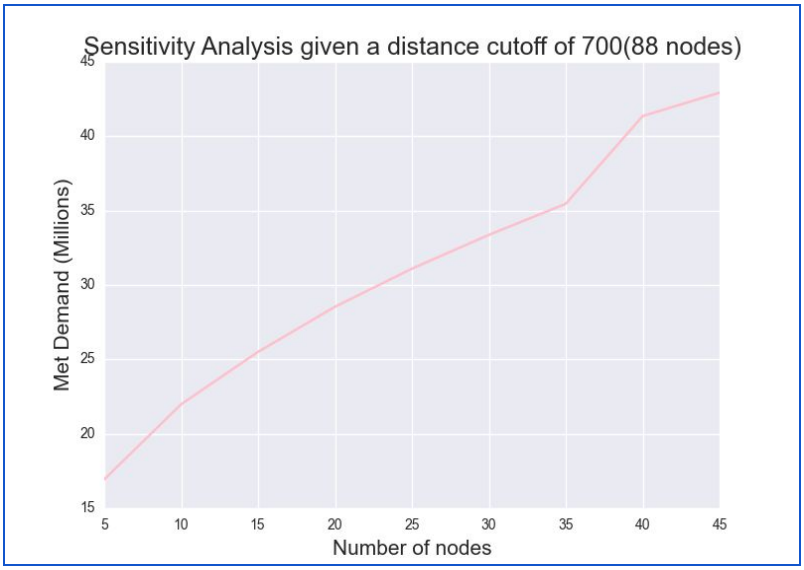


Figure 4.b. Demand in millions vs. number of nodes plot for 88 node network.



While exploring our sensitivity analysis visualisations above, for both networks (49 & 88 nodes) it is expected that the higher is the number of visited nodes, the higher would be the total demand met over the network while there is no obvious trend regarding the optimal distance cutoff especially for the 88 nodes network as the overall demand met oscillates between 22,300,000 And 21,850,000 when the distance cutoff varies between 100 and 2000 and the number of visited nodes is fixed to 10. Regarding the 49 nodes network the demand drops significantly from 137,000,000 to 135,500,000 at a cutoff of 500, then stays at the same low level while we increase the cutoff gradually to 2000.

Conclusions

Lagrangian Relaxation is great way of solving problems of maximal coverage for these vast networks. Some of the advantages we realized with LR are:

1. Computationally inexpensive in terms of time & memory utilization
2. Flexible in terms of addition & deletion of constraints
3. No reliance on specific assumptions
4. Flexible in terms of scalability of demand nodes
5. Always produces a good enough solution within the Upper & Lower Bounds

However, we would exercise caution before we apply this technique to situations where we need the best solution. Examples could be an Ideal network for medicine plants & pharmacy outlets or cases where higher precision is extremely important.