# Computational Experience with the Maximal Location Covering Problem


## Abstract

On this paper, you can find heuristic approaches applied to the Maximal Covering Location Problems; this project could be useful to decision makers that want to generate a solution and try to improve a given solution as well in a short amount of time.


## 1 Introduction

Facility Location is a branch of Operations Research. This category of combinatorial optimization problems often deals with problems that seek to select the placement of a facility (often, from a given list of possibilities) that meets the best of certain constraints. These problems often consists of minimizing the total weighted distances from supplies and customers, and weights are representative of the difficulty of transporting materials.

Then, we have the Maximal Covering Location Problem. This problem is a derivative of Facility Location sort; however, this problems seeks to maximize the customers covered by a given set of facilities,instead of minimizing the distance between customers and supplies.

The Maximal Covering Location Problem (MCLP), is a classic problem in location analysis with applications in a good number of fields, such as health care, emergency planning, ecology, statistical classification, homeland security, etc.

The objective of the MCLP is to maximize the population covered by a given set of facilities; however, this could work as an analogy to another problems that involve "supply" and "demand" points to maximize the demand as much as possible.

Formally, this problem is described in Church and ReVelle 1974 as follows: "The maximal covering location problem seeks the maximum population which can be served within a stated service distance or time given a limited number of facilities."

In this work, MCLP is analysed and solved on random instances generated within a certain range, by an implemented instance generator. A Constructive Heuristic approach was used to generate a solution, and a Local Search algorithm was implemented to try to improve the feasible solution generated by the constructive. Overall, the effectiveness and computational time of the implemented algorithms on small, medium and large instances were analysed. More details are given in the following sections.

## 2 Problem description - Mathematical model

According to the model defined on Church and ReVelle 1974, this problem is defined on a network of nodes and arcs.

A mathematical formulation of this problem can be stated as Eq. 1:

$$max\ z = \sum_{i \in I} y_i \tag{1}$$

subject to:

$$x_j \in (0,1), j \in J \tag{2}$$

$$y_i \in (0,1), i \in I \tag{3}$$

$$\sum_{j \in J} x_j = S \tag{4}$$

$$\sum_{j \in N_i} x_j \geq y_i, N_i = \{j \in J : d_{ij} \leq r\} \tag{5}$$

where

- $x_j$ : denotes that the $j$-th facility site is selected (1) or not (0). Boolean variable.
- $y_i$ : denotes that if the demand point i is covered by any facility site within radius $r$ (1) or not (0). Boolean variable.
- $I$ : the set of demand or population points. Array of indexes referencing the coordinates of each demand point.
- $J$ : the set of facility candidate sites. Array of indexes referencing the supply points.
- $d(i,j)$ : the Euclidean distance from a demand point $i$ to facility site $j$. 2-D matrix.
- $S$ : the number of desired facilities to be selected from the candidate sites $J$. Integer variable.
- $r$ : Maximal distance where a demand point from set $I$ is covered by a facility site from set $J$. Integer variable.

The objective is to maximize the number of demand points covered within a desired service distance, as depicted in 1. A selection criteria for a site is described in restriction 2, where as restriction 3 alludes to all the demands point covered within a specified radius. Ideally, the sum of the selected sites must be equal to the desired by a decision maker as stated on restriction 4; however, it's possible that all the demand/population points are covered before this restriction is satisfied. Restriction 5 indicates that the distance between covered nodes by the selected sites must be under the specified radius from the user.

## 2.1 Example

The scatter plot shown in the figure 1 was generated by an instance generator coded in the programming language Python. The range of the population points are within a range of 5-500 units, and the potential candidate sites were generated inside the convex hull of the demand points. There are exactly 500 demand points, and 20 candidate sites. The values were generated randomly.

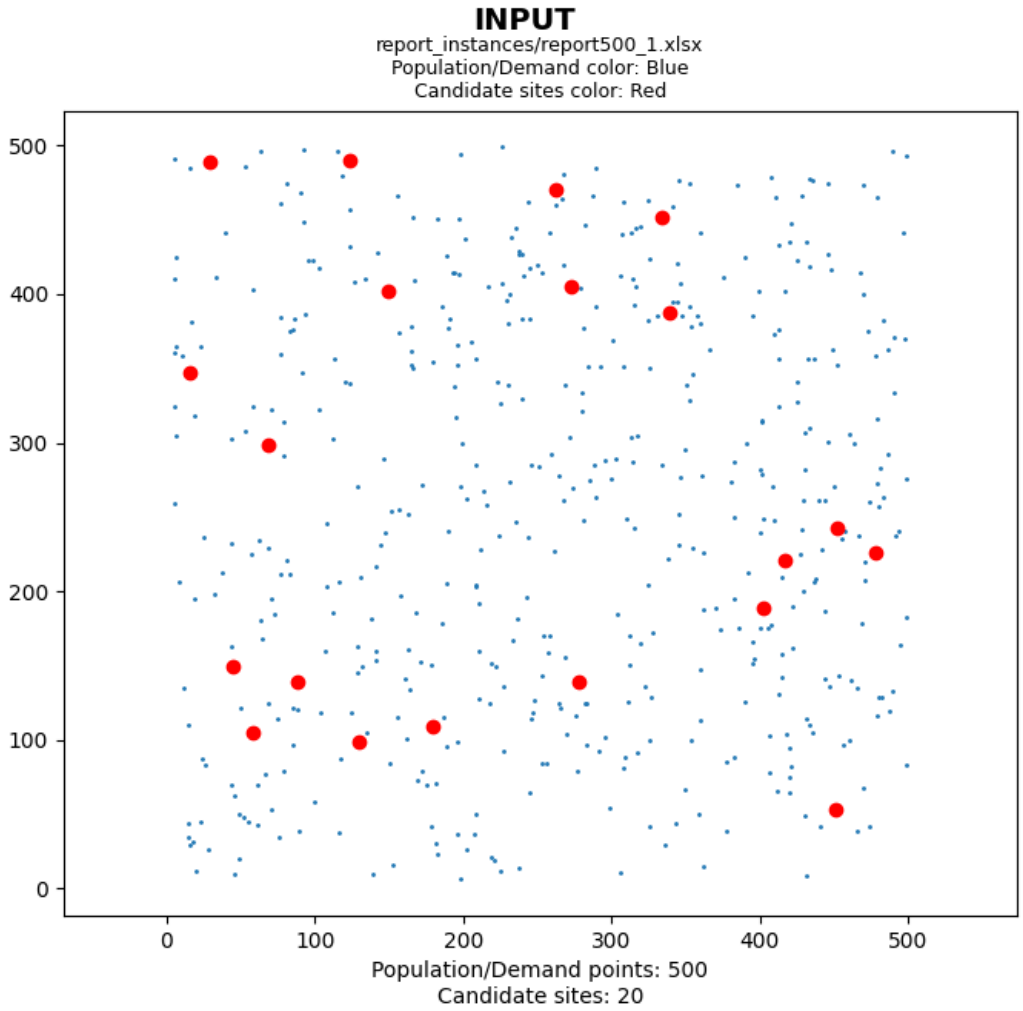


**Fig. 1:** Random instance of population and candidate sites points

As shown, the population points are illustrated in the scatter plot in blue, and the potential candidate sites are filled with red.

Once the input is succesfully computed, there are a few ways to figure out a first feasible solution. As seen in Church and ReVelle 1974, a first heuristic approach called Greedy Adding (GA) was considered. This approach starts with an empty solution, and starts to iterate each candidate site until the one with maximal covered population is found; then, this site is added to the solution. It keeps iterating until either the $S$ facilities have been selected or all the population is covered. However, there is a problem with this approach: the time complexity is denoted as $O(n\ logn)$, which means that it's not efficient for big instances.

Instead of using the Greedy Adding algorithm proposed by these guys, a Constructive Heuristic approach was implemented for generating first a feasible solution that meet the constraints. Then, a Local Search move approach tries to improve this solution. These heuristics are discused in more detail on the following sections.

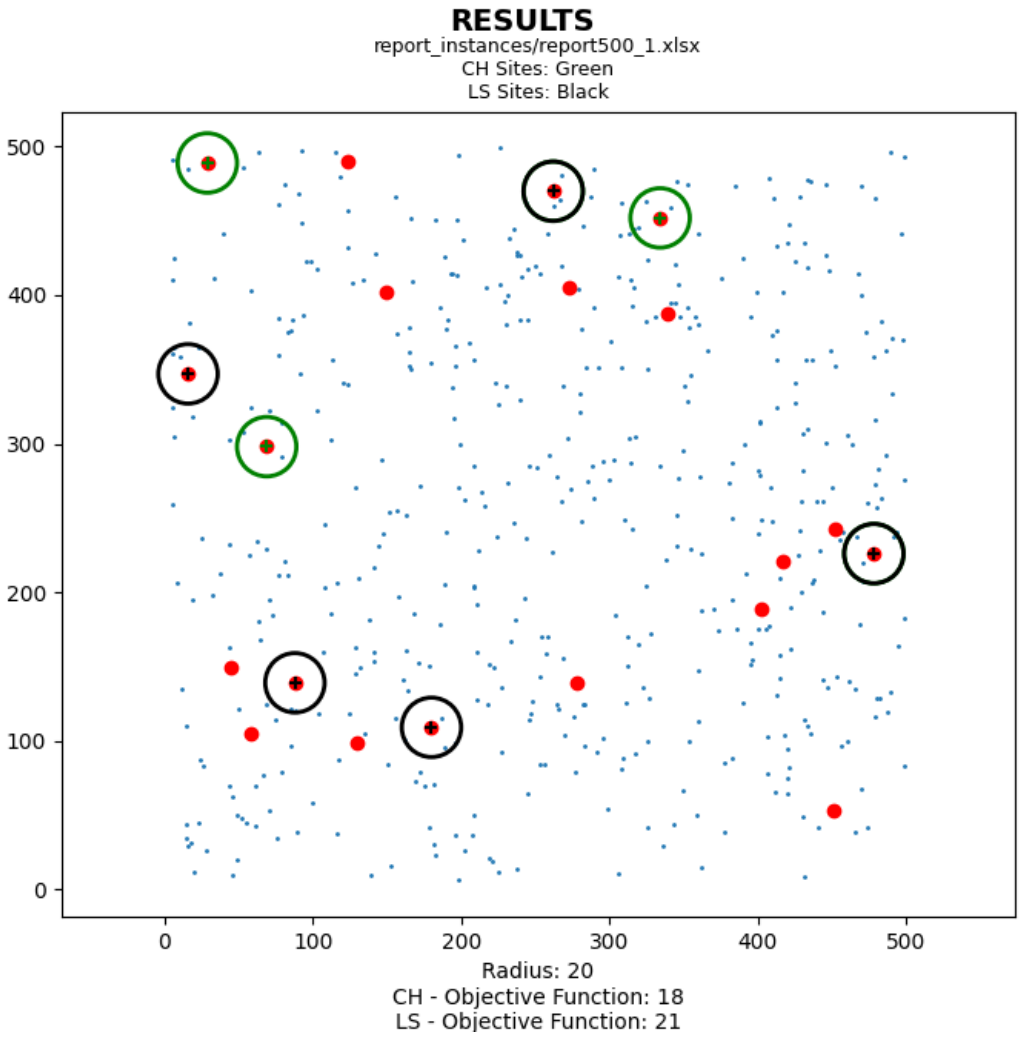


**Fig. 2:** Random instance of population and candidate sites points

The scatter plot shown in the figure 2 illustrates the output of the applied heuristics mentioned previously. The sites and population covered of the Constructive Heuristic approach are marked with green circles, and black circles illustrate the ones covered by Local Search.

# 3 Heuristics

## 3.1 Description of Constructive Heuristic

By definition, a constructive heuristic is a type of heuristic that starts with an empty solution and keeps extending with each iteration until a feasible solution is obtained.

The implementation of the Constructive Heuristic made for this problem receives as input the demand points $I$, the potential candidate sites $J$, the desired sites to be selected $S$ and the

*radius* of coverage. The Constructive Heuristic implemented for the MCLP is described in the Alg. 1.

**Algorithm 1:** Constructive Heuristic

**Input:**

- $I \leftarrow [1, 2, 3...i]$
- $J \leftarrow [1, 2, 3...j]$
- $S \leftarrow int\ value$
- $radius \leftarrow int\ value$

**Output:**

- $selected\_sites \leftarrow [1, 2, 3...S]$
- $objective\_function \leftarrow Population\ covered\ by\ selected\_sites$

**Pseudocode:**

$selected\_sites \leftarrow []$;
$population\_size \leftarrow length(I)$;

**for** *site* **in** $J$ **do**
    $current\_covered\_nodes \leftarrow Compute\ covered\ nodes(selected\_sites)$;
    $objective\_function = length(current\_covered\_nodes)$;
    $r1 = covered\_nodes(under\ radius(site)) > 0$;
    $r2 = length(selected\_sites) < S$;
    $r3 = objective\_function < population\_size$;
    **if** *(r1 and r2 and r3) == True* **then**
        *Append site to selected_sites*;
    **else if** *r1 == False* **then**
        *pass*;
    **else**
        *return selected_sites, objective_function*;
        *break for*;
**end**

### 3.2 Description of Local Search

Before describing the move of the Local Search (LS) algorithm used on this problem, it's important to know what a Local Search is; the main thing that differs between Local Search and Constructive heuristics is that Local Search needs a starting solution, whereas Constructive heursitics starts with a empty solution and build it from scratch.

In general, the objective of Local Search moves is to improve a given solution. Best-found strategy was used in the LS implementation for the MCLP, described in the following algo-

rithm.

**Algorithm 2:** Local Search heuristic

**Input:**

- $I \leftarrow [1, 2, 3...i]$
- $J \leftarrow [1, 2, 3...j]$
- $S \leftarrow int\ value$
- $radius \leftarrow int\ value$

**Output:**

- $selected\_sites \leftarrow [1, 2, 3...S]$
- $objective\_function \leftarrow Population\ covered\ by\ selected\_sites$

**Pseudocode:**

$selected\_sites \leftarrow []$;
$population\_size \leftarrow length(I)$;
**for** *site* **in** $J$ **do**
  $current\_covered\_nodes \leftarrow Compute\ covered\ nodes(selected\_sites)$;
  $objective\_function = length(current\_covered\_nodes)$;
  $r1 = covered\_nodes(under\ radius(site)) > 0$;
  $r2 = length(selected\_sites) < S$;
  $r3 = objective\_function < population\_size$;
  **if** *(r1 and r2 and r3) == True* **then**
    $Append\ site\ to\ selected\_sites$;
  **else if** *r1 == False* **then**
    *pass*;
  **else**
    $return\ selected\_sites,\ objective\_function$;
    $break\ for$;
**end**

# 4 Experiments

Conclusions show readers the value of your completely developed argument or thoroughly answered question. Consider the conclusion from the reader's perspective. At the end of a paper, a reader wants to know how to benefit from the work you accomplished in your paper.

# 5 Conclusions

Conclusions show readers the value of your completely developed argument or thoroughly answered question. Consider the conclusion from the reader's perspective. At the end of a paper, a reader wants to know how to benefit from the work you accomplished in your paper.

# A Supporting information

The purpose of the supporting information is to enable authors to provide and archive supporting information such as data tables, method information, figures, video, or computer software, in digital formats so that other scientists can use it.