Studies on Various Maximal Covering Location Problems using Genetic and ABC Algorithms

A Thesis Submitted in Fulfillment of
The Requirements for the Award of The Degree
OF

MASTER OF SCIENCE

IN
Computer Science

By
Subhrajit Das
Registration: 2080002 of 2021-2022
Roll: 90/MCS No: 210015

Dissertation- II (MCS-DISS-421)

Thesis Supervisors:

Prof. Priya Ranjan Sinha Mahapatra
Department of Computer Science and Engineering, University of Kalyani
Dr. Soumen Atta
Assistant Professor, Centre For Information Technologies and Applied Mathematics, University of Nova Gorica, Slovenia

June, 2023

CERTIFICATE

This is to certify that this report on **"Studies on Various Maximal Covering Location Problems using Genetic and ABC Algorithms"** was based on course-work of M.Sc. carried out by Subhrajit Das under the supervision of Prof. Priya Ranjan Sinha Mahapatra, Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia and Dr. Soumen Atta, Assistant Professor, Centre For Information Technologies and Applied Mathematics, University of Nova Gorica, Slovenia.

The content of this report, in full or part, has not been submitted to any other university or institution for the award of any degree or diploma.

Signature:
Prof. Priya Ranjan Sinha Mahapatra
Department of Computer Science and Engineering
University of Kalyani, Kalyani, Nadia

Signature:
Dr. Soumen Atta
Assistant Professor
Centre For Information Technologies and Applied Mathematics
University of Nova Gorica, Slovenia

Abstract

This thesis report comprises two chapters. The first chapter presents the implementation and experimental results of a Genetic Algorithm with Local Refinement for the Maximal Covering Location Problem on SJC-datasets (SJC324, SJC402, SJC500, SJC708, and SJC818). The proposed method matches the best-known solutions for 60 out of 82 instances, demonstrating its promising performance. The second chapter proposes an Artificial Bee Colony algorithm with Regional Facility Enhancement for the solution of NP-Hard Probabilistic Maximal Covering Location-Allocation Problem (PMCLAP). The proposed method aims to improve the solution quality and convergence speed by applying a regional facility enhancement procedure. The PMCLAP problem also involves allocating customers to suitable facilities to achieve optimality, and several strategies are suggested for this sub-problem. The effectiveness and efficiency of the proposed method are discussed and illustrated by solving several instances on three data-sets of different sizes: 30-node, 324-node, and 818-node networks with waiting queue size constraint and waiting time constraint. The proposed heuristic method shows promising performance on the 30 & 818-Node data-sets, where it matches the optimal solutions obtained by CPLEX in most cases. However, on the 324-Node data-set, there is a noticeable gap between the heuristic and optimal solutions. Overall, the heuristic method attains 50% of the optimal solutions achieved by CPLEX, with an average gap of 0.09% for the standard instances tested. The quality of the heuristic solutions depends largely on the customer allocation strategy. Future research will focus on developing improved allocation strategies to enhance the performance of the heuristic method.

Acknowledgements

I am deeply grateful to my supervisors, Prof. Priya Ranjan Sinha Mahapatra from the Computer Science and Engineering Department at the University of Kalyani, and Dr. Soumen Atta from the School of Engineering and Management at the University of Nova Gorica. Their guidance, support, and motivation have been invaluable in shaping my research and propelling me to new achievements.

I also extend my thanks to the Computer Science and Engineering Department at the University of Kalyani for granting me access to essential laboratory resources that played a vital role in the successful completion of this project. The unwavering support and encouragement from the faculty and staff in the department have been a constant source of inspiration.

Lastly, I want to express my heartfelt appreciation to my family and friends who have been with me throughout my academic journey. Their love, encouragement, and unwavering support have been the solid foundation of my accomplishments.

Contents

Chapter 1

Implementation of Maximal Covering Location Problem using Genetic Algorithm with Local Refinement

1.1 Background

The Maximal Covering Location Problem (MCLP) is a extensively researched issue in the field of facility location analysis, first proposed by Church and ReVelle [1974].The objective of this problem is to identify the best location of facilities in a specific area that can serve a collection of demand points with the highest coverage. In simpler terms, we can explain it like: we are directed to open k facilities among N cities having respective population such that maximum possible population is served.

The origins of the MCLP can be traced back to the 1960s when researchers began exploring location-allocation problems. Since then, it has gained considerable attention due to its relevance in practical applications such as facility location planning, service network design, and resource allocation.

In the MCLP, each facility is assumed to have a predetermined coverage range or service radius within which it can serve the demand points. The aim is to choose a set of possible candidate location for facilities among the demand points that can cover the maximum number of demand points with the condition that every demand point has a minimum of one facility serving it. The problem exhibits complexity owing to various factors, namely the location of demand points, the coverage area of facilities, and potential limitations on resources or capacity. Real-world instances of the MCLP often involve large-scale geographic regions with numerous demand points, resulting in high computational difficulty for searching precise solutions in feasible time periods.

For addressing the MCLP, researchers have proposed diverse mathematical models, optimization algorithms, and heuristic approaches. These methods aim to find near-optimal solutions by striking a balance between computational efficiency and solution quality. Meta-heuristic algorithms like particle swarm optimization, simulated annealing, and genetic algorithms have shown promise in solving larger instances of the problem. Given its practical significance in emergency service planning, healthcare facility location, and supply chain management, the MCLP continues to attract research attention. The ultimate goal is to develop innovative algorithms and solution approaches that can effectively handle the complexities of the problem and provide decision-makers with reliable tools for making optimal facility location decisions.

1.2 Problem Definition

As defined by Church and ReVelle [1974], assume a set P consisting m points in a two-dimensional plane, denoting cities or customers. Every point $p_j \in P$, where j ranges between {1, 2, 3, ..., m}, having non-negative population or demand, indicating the weight of the corresponding point.

Each facility's area of service, or area of coverage, is circular in nature and has a constant radius r, indicating the service distance. The center of the circle with service radius r denotes the location of the facility. We are given k number of facilities to be opened, where k ranges from 1 to m, and the set $\{c_1, c_2, ..., c_k\}$ which represents the centers for the k facilities. Notably, the potential locations for every facility are limited within customers' locations, implying $c_i \in P$ for i ranging from 1 to k.

Iff $d(p_j, c_i)$, the euclidean distance between p_j and c_i, is at most rA customer or demand point $p_j \in P$ is considered to be in the coverage of a facility circle having center at c_i, where i ranges from 1 to k. In-case a customer or demand point is within the service radius of multiple facilities then any of the feasible facility can serve the customer. But, a customer can be served by at-most one facility. Maximizing the total sum of demands of the customers within the coverage by finding the possible candidate locations of k facilities is the objective of MCLP.

Consider the set $P = \{p_1, p_2, ..., p_m\}$ as the set of customers, the possible candidate facility location set $I = \{c_1, c_2, ..., c_m\}$, and the demand of customer p_j denoted by f_j. We consider r as the service distance or radius of for all the candidate facilities. And we assume k is the number of facilities to be opened. If the customer $p_j \in P$ can be served by a facility located at $c_i \in I$ we set $a_{ij} = 1$, else we set $a_{ij} = 0$. Additionally, if customer p_j is served we set $x_j = 1$, else we set $x_j = 0$; $y_i = 1$ infers a facility to be located at site $c_i \in I$, else $y_i is 0$.

We consider the following objective function of MCLP as defined in Atta et al. [2018] v(MCLP), is as follows:

$$v(\text{MCLP}) = \max_{p_j \in P} f_j x_j \tag{1}$$

subject to the constraints:

$$\sum_{c_i \in I} a_{ij} y_i - x_j \geq 0, \quad p_j \in P \tag{2}$$

$$\sum_{c_i \in I} y_i = k \tag{3}$$

$$x_j \in \{0, 1\}, \quad p_j \in P \tag{4}$$

$$y_i \in \{0, 1\}, \quad c_i \in I \tag{5}$$

The total covered demand is represented by the objective function (1) in the formulation, and the aim is to achieve highest possible total covered demand. A demand point $p_j \in P$ is covered or served iff there exists at least one facility $c_i \in I$ such that $d(p_j, c_i) \leq r$ is stated by constraint (2). The count of facilities to exactly k is limited by constraint (3). The binary nature that is either 0 or 1 of the decision variables for the problem is enforced by constraints (4) and (5).

For this implementation, we assume the customers' locations as the possible candidate facility locations. Thus, the sets I and P are similar or comparable for the problem implemented in this report.

.

1.3 Procedure of Implementation

For implementing the MCLP using GA with Local Refinement, strategy similar to Atta et al. [2018] is taken into consideration. The flow of execution of the strategy is depicted in the flowchart fig. 1.1. In the following subsections, brief description of each step is

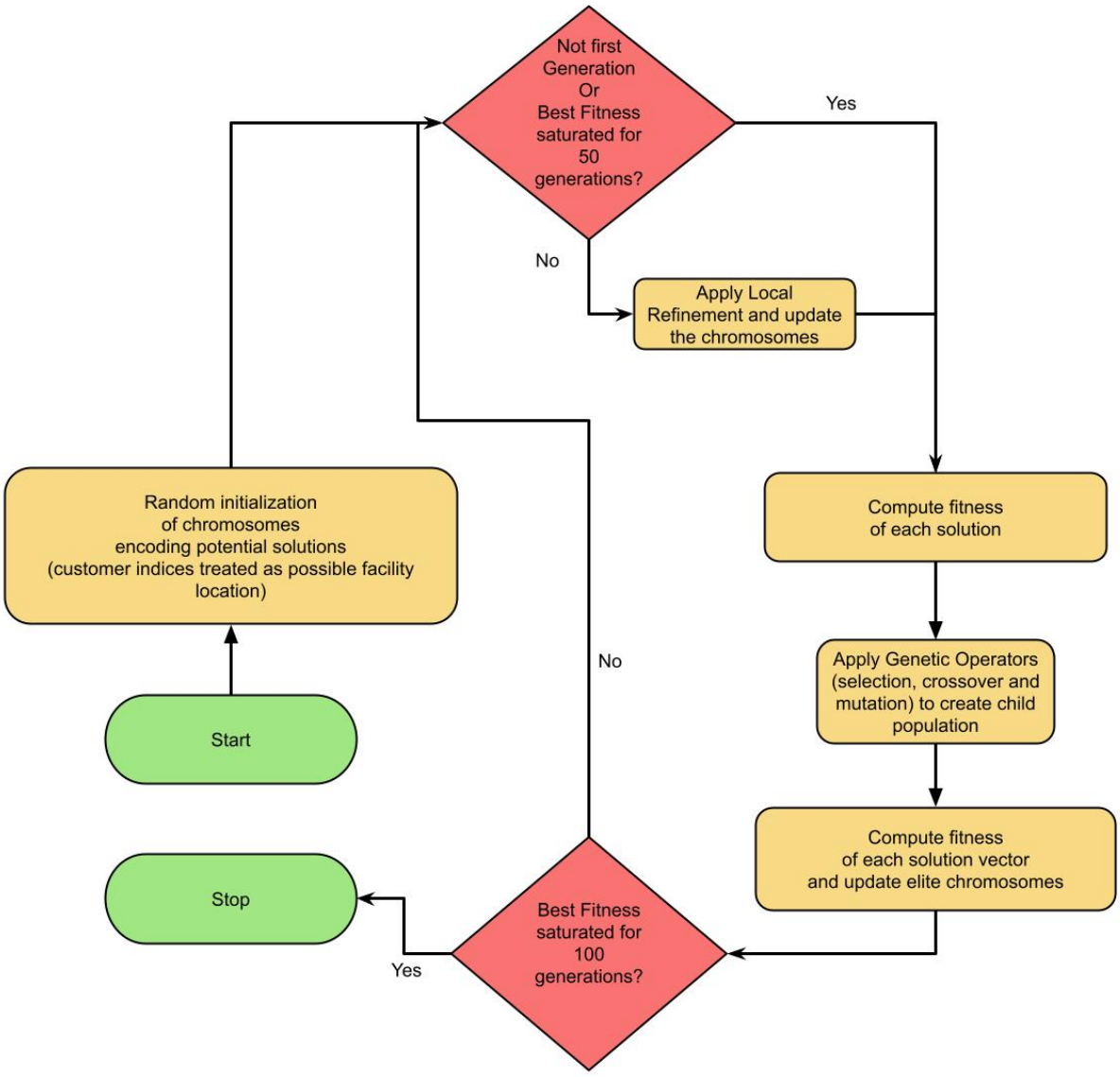

Figure 1.1: Flowchart for solving MCLP based on the Genetic Algorithm strategy

provided.

1.3.1 Encoding of Chromosomes

In genetic algorithm, possible candidate solutions are encoded as strings called as chromosomes. For this MCLP problem, chromosomes are made up of string of integer indices representing the possible candidate facility locations to be opened from the set of customers P = {p_1, p_2, p_3, ... , p_m}, similar strategy to Atta et al. [2018]. Therefore, a chromosome is an integer string $\{t_1, t_2, t_3, ..., t_k\}$ of length k, where k denotes the count of facilities to be opened, and every $t_i \in t$ refers to a customer index where a facility has been opened. Since the potential facility locations are restricted to the customers themselves, each element t_i corresponds to a customer location. Let us assume some values: $m = 324$ and $k = 10$, a string of chromosome $\{142, 52, 98, 79, 101, 320, 141, 65, 55, 10\}$ implies that customers p_{142}, p_{52}, p_{98}, p_{79}, p_{101}, p_{320}, p_{141}, p_{65}, p_{55}, and p_{10} are selected as possible facility locations to be opened according to the encoded solution in the chromosome.

1.3.2 Initialization of Population

Random initialization of the initial population, consisting of P chromosomes are done, where P denotes the size of population. For the initial population, each of the chromosome is randomly generated by choosing k indices from the set $\{1, 2, 3, ..., m\}$. The pseudo-code implemented in MATLAB is shown in 1.

Algorithm 1 initialize

```
// Inputs: P, K, m
// Output: population (PxK vector)
// P is the population size
// K is the number of facilities to be opened
// m is the size of customers
1: population = zeros(P, K, 1)
2: for i = 1 to P do
3:     population(i, :, 1) = randperm(nrows, K)
```

1.3.3 Computation of Fitness

The goodness or quality of a encoded chromosome for MCLP is indicated by the fitness. The fitness computation for MCLP is done as:

$$\frac{\sum_{i=1}^{k} \text{Demand of customers served by facility } i}{\text{Total demand of all customers}} \times 100\%$$

where, k is the number of facilities opened. And aim is to maximize this fitness value. So, each chromosome encodes fitness as the coverage of the solution.

1.3.4 Genetic Operators

Selection, crossover and mutation are the three genetic operators involved to create the population of next generation, Each of the three genetic operators are described in the following subsections.

Selection

The process of generating a mating pool from the population is called as selection. The chromosomes that undergo the selection procedure in the mating pool qualify for the subsequent operation crossover. For this selection procedure, a popular selection strategy called binary tournament selection Goldberg [1989] is used. The binary tournament selection strategy used for selection of chromosomes of MCLP is described in the pseudo-code 2.

Algorithm 2 selection

```
// Inputs: population, P, K
// Output: mating_pool (PxK vector)
mating_pool = zeros(P,K)
for i = 1 to P do
    p1 = randi([1, P], 1)
    p2 = randi([1, P], 1)
    if fitness(p1) > fitness(p2) then
        mating_pool(i,:) = population(p1,:)
    else
        mating_pool(i,:) = population(p2,:)
```

Crossover

Crossover operation is a procedure that entails the transfer or exchange of information between two parent chromosomes with the aim of producing two offspring chromosome solutions that are distinct from their progenitors. There are several crossover strategies in the literature, for our implementation single-point crossover strategy is utilized. The strategy implemented is shown in the pseudo-code 3.

Algorithm 3 crossover

// Inputs: mating_pool, P, K
// Output: C (PxK vector)

$x = 1$
$off1 = zeros(1, K)$
$off2 = zeros(1, K)$
for $i = 1$ to $(P/2)$ **do**
    $r = rand$
    $p1 = randi([1, P], 1)$
    $p2 = randi([1, P], 1)$
    if $(r \leq crossprob)$ **then**
        $crosspoint = randi([1, K], 1)$
        for $m = 1$ to $crosspoint$ **do**
            $off1(1, m) = M(p1, m)$
            $off2(1, m) = M(p2, m)$
        for $m = crosspoint + 1$ to K **do**
            $off1(1, m) = M(p2, m)$
            $off2(1, m) = M(p1, m)$
    else
        for $m = 1$ to K **do**
            $off1(1, m) = M(p1, m)$
            $off2(1, m) = M(p2, m)$
    $C(x, :) = off1(1, :)$
    $C(x + 1, :) = off2(1, :)$
    $x = x + 2$
    $off1 = []$
    $off2 = []$

The crossover operation is governed by a fixed crossover probability μ_c, and it is performed $P/2$ times to produce P offspring chromosome solutions. In this implementation, μ_c is fixed to 0.9 for all generations.

Mutation

Slight alteration of a single parent chromosome is known as mutation. The mutation strategy implemented is shown in 4:

Algorithm 4 mutation

```
// Inputs: C, P, K, m
// m is the number of customers
// Output: updated pool T
neighbourSize = m/10
// For 10% neighbours
T = C
for i = 1 to P do
    for j = 1 to K do
        r = rand
        if (r ≤ μ_m) then
            neighbour = getClosestNeighbours(C(i, j), neighbourSize)
// getClosestNeighbours returns the closest neighbourSize neighbour customer indices
//     from the facility C(i,j)
            mutated = randsample(neighbour, 1)
            T(i, j) = mutated
```

Initially the μ_m is set to 0.01 and it is kept unchanged till local refinement procedure is applied on the population, ensuring mutation process in the early convergence stage through local improvements does not result in substantial disturbances to the chromosomes. Once the need for local refinement diminishes, the mutation probability μ_m can be appropriately calibrated further and in our implementation it is adjusted to 0.8.

These steps constitute the main procedure of the proposed Genetic Algorithm based solving approach for Maximal Covering Location Allocation Problem. And the subsequent sections, further details are discussed for performance elevation of the proposed method.

Local Refinement

Following the mutation procedure, a local refinement process is applied to each of the solution (chromosome) of the pool. The refinement procedure is carried out in the following way: Initially clusters of customers are created around facilities. The assignment of each

customer p_i is based on their proximity to a facility c_j. Each facility c_j has a cluster $clst_j$ of customers that are assigned to it(Jain et al. [1999]; Mukhopadhyay et al. [2015]). Updating each facility c_j with c_t such that,

$$t = \arg\min_{p_i \in clst_j} \sum_{p_l \in clst_j} fld(p_i, p_l). \tag{1.1}$$

where the chromosome encodes each facility c_j

Therefore, the point with the lowest weighted sum of distances to other points in its cluster is chosen as the new location for each facility. By doing this, the facility locations become more aligned with points that are centrally located and have more demand, which results in a faster increase in the total possible coverage. This refinement helps the algorithm converge faster when it is applied in the initial stages. Moreover, it is significant to mention that this is true for most of the cases, the farther the customer and facility are from each other, the worse the service quality becomes. Hence, this improvement method also tries to offer superior service to customers close to the facility than to those who are more distant. If there is no change in the best fitness function for 50 consecutive generations, the local refinement procedure is discontinued to allow free evolution of the chromosomes.

Elitism

Elitism preserves the best quality chromosome obtained till date. This is a way to protect quality solutions from being lost due to the stochastic nature of selection, crossover, and mutation genetic operators . The parent population and child populations for a given generation are combined, and then P solutions having highest fitness are chosen to proceed for the next generation.

Termination Criterion

The algorithm iterates across generations the process of computing fitness, selecting, crossing over, mutating, refining locally, and applying elitism. The local refinement of chromosomes keeps going until 50 generations have the same best fitness value. Then, the local refinement is discontinued, and the chromosomes evolve freely. The loop keeps going until no improvement in the best fitness value is seen for the last 100 generations. The

best solution, corresponding to the highest coverage value, is what the GA outputs.

1.4 Experimental Results

For assessing the quality of the Genetic Algorithm with local refinement procedure, this section provides the experimental results and details. The implementation was coded in Matlab™ version: R2021a. Computational experiments of the data-set used were done on machines configured with an Intel i5™ processor clocked at 2.5 GHz frequency needing less than 500 Megabytes of RAM memory. Tables 1.1 - 1.5 shows the experimental results for various data-sets and their instances. Experiment incorporates five data-sets: SJC324, SJC402, SJC500, SJC708, and SJC818. These experimental result achieved by the implementation looks promising.

Table 1.1: Experimental Results with GA with refinement-based solving for SJC324 MCLP

n	p	S	Cov.(%)	Gap (%)	Time(s)
324	1	800	**44.94**	**0**	1.64
324	2	800	**72.33**	**0**	2.47
324	3	800	**95.49**	**0**	3.43
324	4	800	**99.62**	**0**	5.66
324	5	800	**100**	**0**	2.95
324	1	1200	**81.73**	**0**	4.62
324	2	1200	**95.08**	**0**	4.18
324	3	1200	**100**	**0**	3.17
324	1	1600	**99.76**	**0**	5.65
324	2	1600	**100**	**0**	3.77

Table 1.2: Experimental Results with GA with refinement-based solving for SJC402 MCLP

n	p	S	Cov.(%)	Gap (%)	Time(s)
402	1	800	**41.01**	**0**	1.58
402	2	800	**70.94**	**0**	2.71
402	3	800	**91.9**	**0**	4.65
402	4	800	97.85	0.11	5.51
402	5	800	**99.91**	**0**	4.89
402	6	800	**100**	**0**	4.03
402	1	1200	**66.36**	**0**	4.16
402	2	1200	**92.79**	**0**	4.95
402	3	1200	**100**	**0**	4.53
402	1	1600	**96.58**	**0**	7.37
402	2	1600	**100**	**0**	5.53

Table 1.3: Experimental Results with GA with refinement-based solving for SJC500 MCLP

n	p	S	Cov.(%)	Gap (%)	Time(s)
500	1	800	**40.31**	**0**	1.84
500	2	800	**63.2**	**0**	3.41
500	3	800	**79.82**	**0**	4.81
500	4	800	90.18	0.11	8.03
500	5	800	**95.7**	**0**	15.83
500	6	800	**99.08**	**0**	17.20
500	7	800	**99.92**	**0**	11.87
500	8	800	99.97	0.03	5.426376
500	1	1200	**54.43**	**0**	3.20
500	2	1200	**91.69**	**0**	6.52
500	3	1200	**98.41**	**0**	7.05
500	1	1600	**75.12**	**0**	6.40
500	2	1600	**99.8**	**0**	7.74
500	3	1600	**100**	**0**	6.14

Table 1.4: Experimental Results with GA with refinement-based solving for SJC708 MCLP

n	p	S	Cov.(%)	Gap (%)	Time(s)
708	1	800	**34.69**	**0**	2.58
708	2	800	**55**	**0**	4.65
708	3	800	**71.4**	**0**	6.29
708	4	800	**84.07**	**0**	17.57
708	5	800	**88.81**	**0**	35.01
708	6	800	92.69	0.33	30.68
708	7	800	94.75	0.95	24.42
708	8	800	**97.83**	**0**	26.25
708	9	800	98.5	0.6	14.48
708	10	800	98.95	1.4	20.93
708	11	800	**100**	**0**	14.828
708	1	1200	**48**	**0**	4.52
708	2	1200	**84.23**	**0**	9.10
708	3	1200	**92.68**	**0**	12.58
708	4	1200	**98.73**	**0**	27.78
708	5	1200	99.66	0.13	20.04
708	6	1200	**100**	**0**	10.31
708	1	1600	**69.56**	**0**	9.41
708	2	1600	**96.59**	**0**	18.10
708	3	1600	98.59	0.15	10.36
708	4	1600	**100**	**0**	8.66

Table 1.5: Experimental Results with GA with refinement-based solving for SJC818 MCLP

n	p	S	Cov. (%)	Gap (%)	Time (s)
818	1	800	**28.77**	**0**	2.67
818	2	800	**45.62**	**0**	10.84
818	3	800	**60.02**	**0**	15.53
818	4	800	73.10	0.36	7.64
818	5	800	83.21	0.89	21.21
818	6	800	87.49	1.33	24.52
818	7	800	90.19	2.15	24.96
818	8	800	94.06	1.29	35.48
818	9	800	96.59	0.77	37.87
818	10	800	97.67	0.88	37.47
818	11	800	98.93	0.81	29.14
818	12	800	99.14	0.67	26.68
818	13	800	**99.98**	**0**	34.59
818	14	800	**100**	**0**	15.56
818	1	1200	**39.81**	**0**	7.98
818	2	1200	**69.56**	**0**	16.68
818	3	1200	**86.43**	**0**	23
818	4	1200	92.46	0.21	27.09
818	5	1200	97.35	0.40	26.63
818	6	1200	99.59	0.30	29.47
818	7	1200	99.96	0.04	11.74
818	1	1600	**57.69**	**0**	10.12
818	2	1600	**84.5**	**0**	15.12
818	3	1600	**94.87**	**0**	21.76
818	4	1600	**98.95**	**0**	22.39
818	5	1600	**100**	**0**	10.91

1.5 Conclusion

In conclusion, this study implemented the Maximal Coverage Location Problem using Genetic Algorithm with local refinement strategy and showed the experimental results achieved by the implementation. It shows promising results for 60 out of 82 instances for the SJC324, SJC402, SJC708 and SJC818 data-sets, in terms of both achieving near-optimal benchmark results and computational time. However, for some instances the benchmark results were missed by a small margin compared to the benchmark result, however it beats some of the existing models in terms of computational time by a multi-fold times. Future work will focus on making appropriate changes and improvements to achieve faster computational time and also giving overview of other evolutionary strategies to

solve this problem. Overall, this study provides insights into implementing the MCLP problem and opens up avenues for further research in this area.

Chapter 2

Solving Probabilistic Maximal Covering Location Allocation Problem using Artificial Bee Colony Algorithm with Regional Facility Enhancement

2.1 Introduction

Introduced by Marianov and Serra [1998], the probabilistic maximal covering location-allocation problem (PMCLAP) is a NP-hard optimization problem in the area of operations research which is a modified and constraints imposed on the maximal covering location problem (MCLP) proposed by Church and ReVelle [1974] to achieve minimum service quality. The problem can be practically applied into enormous use-cases for example locating places for first-aid centers, hospitals, fire stations, locating electric vehicle (EV) charging stations, stores of fast food chains, placing ATMs, etc; and even when we need to open K facilities in a country of N cities having respective population to serve while maintaining minimum service quality at each facility.

In recent years, swarm intelligence-based algorithms have shown great potential in solving optimization problems. One of such popular swarm intelligence-based algorithms is the Artificial Bee Colony algorithm (ABC) introduced by Karaboga et al. [2005]. As the PMCLAP problem is also an optimization problem, we have proposed to use the ABC algorithm to solve PMCLAP problem, which incorporates proposed regional facility enhancement strategy for attaining better results with quicker convergence. The swarm-based optimization algorithm Artificial Bee Colony (ABC) is inspired by honeybees foraging behavior. Basically the algorithm is made up of three types of bees: employed bees, onlooker bees, and scout bees. Employed bees perform local search around their current solutions and communicate their findings to onlooker bees, who use this information to select promising solutions. Scout bees explore the search space for new solutions. These three bees procedure is considered as one cycle of iteration. An iteration refers to one complete cycle of the algorithm, which involves generating and evaluating candidate solutions, selecting the best solutions, and updating the search process based on the knowledge gathered from the preceding epochs. The number of iterations in ABC is typically specified as a stopping criterion and can be adjusted based on the problem complexity and required solution accuracy. Algorithm 5 shows the basic process of the ABC Algorithm proposed by Karaboga [2010], and this can also be found in Atta et al. [2022].

ABC algorithms are known for solving various NP-hard problems with near-global optimal result, but they can have a high computation time. Local improvement strategies

Algorithm 5 Basic ABC

for $i = 1$ to N **do**
    Randomly initialize the solution vector $\mathbf{X}_i$;
    Evaluate the nectar of each solution $\mathbf{X}_i$;
    $C_i \leftarrow 0$; // abandonment counter
while stopping criterion is not met **do**
    // Employed bees phase
    for $i = 1$ to N **do**
        Create a new solution vector $\mathbf{Y}_i$ from the employed bee $\mathbf{X}_i$ using eq (2.9);
        Compute the nectar of $\mathbf{Y}_i$;
        if $f(\mathbf{Y}_i) \geq f(\mathbf{X}_i)$ **then**
            $\mathbf{X}_i \leftarrow \mathbf{Y}_i$;
        else
            $C_i \leftarrow C_i + 1$; // update abandonment counter
    // Onlooker bees phase
    for $j = 1$ to N **do**
        Based on selection probabilities using roulette wheel selection, select a solution $\mathbf{X}_i$;
        Generate a new solution vector $\mathbf{Y}_i$ from the employed bee $\mathbf{X}_i$ using eq. (2.9);
        Compute the nectar of $\mathbf{Y}_i$;
        if $f(\mathbf{Y}_i) \geq f(\mathbf{X}_i)$ **then**
            $\mathbf{X}_i \leftarrow \mathbf{Y}_i$;
        else
$C_i \leftarrow C_i + 1$; // update abandonment counter
    // Scout bees phase
    for $i = 1$ to N **do**
        if $C_i > L$ **then**
            Generate a new solution vector $\mathbf{Y}_i$ randomly;
            $\mathbf{X}_i \leftarrow \mathbf{Y}_i$;
            Compute the nectar of $\mathbf{Y}_i$;
            $C_i \leftarrow 0$;
    Update the best solution $\mathbf{X}_g$ found so far;
return $\mathbf{X}_g$

can be incorporated into the ABC algorithm to improve its performance. In this approach, candidate solutions are encoded for probable facility location indices, and the nectar or objective function is calculated as the sum of demand or population served. A regional facility enhancement strategy is considered to fine-tune food sources of the solution vectors and improve convergence speed. Comparison between the proposed ABC-algorithm technique with regional facility enhancement and the results achieved by the ANLS [Pereira et al., 2015], MS Heuristic [Marianov and Serra, 1998], CS (Clustering Search) [de Assis Corrêa et al., 2007], GRASP [Feo and Resende, 1995], CPLEX [Pereira et al., 2015] model are done with respect to computational time and total demand served. Experimental results show that the suggested technique outperforms the MS Heuristic in most cases in terms of total demand served; outperforms the GRASP in some of the cases in terms of total demand served. While the result achieved by ABC with Regional Facility Enhancement is at par with ANLS and CPLEX is of majority of the cases for 30 & 818-node network, but for 818-node data-sets it beats them in terms of computational time.

The subsequent sections of the chapter are organized as follows: section 2.2 discusses the similar works; mathematical formulation of PMCLAP is described in section 2.3; section 2.4 briefs the suggested ABC technique in detail; the experimental results are reported and discussed in the section 2.5; and at the end report is concluded in section 2.6 .

2.2 Related Works

From the introduction of PMCLAP by Marianov and Serra [1998], various similar versions have been introduced in probabilistic location allocation literature (Marianov and Serra [1998];de Assis Corrêa et al. [2007];de Assis Corrêa et al. [2009];Pereira et al. [2015]). Opening k facilities among N cities is considered in the MCLP introduced by Church and ReVelle [1974]. But this basic MCLP problem doesn't take into account the congestion issues or capacities issue. The PMCLAP was introduced by Marianov and Serra [1998], an modified version of the MCLP imposing least quality on the level of service, which assumes that Poisson distribution is followed by the clients in arrival to the facilities. By counting the the total number of population waiting for service, or by taking into account the waiting for the service, demand at a facility is calculated. In the current work, we are considering the model of PMCLAP introduced by Marianov and Serra [1998]. Several researchers have introduced various strategies to solve the PMCLAP includes Hybrid Heuristics, which merges both the Genetic Algorithm (GA) and the Simulated Annealing (SA) methods like de Assis Corrêa et al. [2007]. Also de Assis Corrêa et al. [2009] proposed a decomposition approach for the PMCLAP. The proposed method decomposes the original problem into a set of subproblems, each of which is then solved using a GA-based algorithm. By combining the solutions of the sub-problems, the solution of the original problem is achieved.The authors of Pereira et al. [2015] introduced a hybrid technique for solving the probabilistic maximal covering location-allocation problem (PMCLAP) which merges simulated annealing (SA) and a genetic algorithm (GA) to enhance the quality of the solutions. The SA method is used to generate initial solutions, while the GA method is used to refine the solutions obtained by SA. The proposed hybrid method was run on several existing benchmark instance datasets and compared with other existing methods, and the computational-results showed that the hybrid method beats the several existing methods in-terms of quality of solution and computational time. introduced by Huang et al. [2022] introduced a particle swarm-optimization (PSO) algorithm for solving the PMCLAP. The proposed method uses a pool of particles for finding the optimal solution. Based on the highest quality solution found so far it updates the velocity and the position of the particles.

2.3 Problem Definition

As proposed by Marianov and Serra [1998], a graph consisting of n nodes in set N is considered for the problem definition where each of the node is assigned with a demand d_i. A service radius r is assigned for all the candidate facilities. The nodes within r units of distance from node i, i.e., the set of location candidates j that can serve customer/client i is considered in the subset N_i. f_i is the contribution in terms of congestion of customer i to the system congestion and is computed as a fraction of the customer's demand. An assumption is made that customers' arrival to the facilities will be followed according to a Poisson distribution with parameter rate μ. The minimum probability of at most

- a waiting queue with b clients, or;
- a waiting time of τ minutes

is defined by the parameter α. For modeling the PMCLAP, two sets of binary variables are defined: one for location and the other for decisions of allocation. Variables y_j are set to 1 if location $j \in N$ is opened, and variables x_{ij} are set to 1 if customer i is served by facility j where $i, j \in N$. The mathematical formulation of the problem is given below, defined in the work Pereira et al. [2015] as follows:

$$maximize \sum_{i \in N} \sum_{j \in N_i} d_i x_{ij} \tag{2.1}$$

subject to

$$\sum_{j \in N_i} x_{ij} \leq 1, \quad i \in N \tag{2.2}$$

$$\sum_{i \in N} y_j = p \tag{2.3}$$

$$x_{ij} \leq y_j, \quad i \in N,\ j \in N \tag{2.4}$$

$$\sum_{i \in N} f_i x_{ij} \leq \mu \sqrt[b+2]{1-\alpha} \quad j \in N \tag{2.5}$$

$$\sum_{i \in N} f_i x_{ij} \leq \mu + \frac{1}{\tau} \ln(1-\alpha), \quad j \in N \tag{2.6}$$

$$y_j, x_{ij} \in \{0,1\}, i \in N, j \in N \tag{2.7}$$

The optimization problem aims to maximize the total demand served, with constraints ensuring the allocation and location of facilities. Specifically, the objective function (2.1) maximizes the total demand/population served, while constraints (2.2) guarantee that at most one facility serves each client, and constraint (2.3) determines the counts of facilities to be opened. Linking the location to allocation variables, by allowing customers to be allocated to an opened facility, is achieved by Constraint (2.4). Constraint (2.5) ensures that facility j has fewer than b clients/customers in the waiting queue with at least probability α, while constraint (2.6) ensures that the waiting time for service at facility j is at most τ minutes with a probability of at least α. The binary nature of the variables is enforced by constraints (2.7). It is worth noting that x_{ij} is set to zero for all $j \notin N_i$.

Constraints (2.5) and (2.6) account for the probabilistic characteristics of the problem. Following the approach of Marianov and Serra [1998], an M/M/1/∞/FIFO queueing system is taken, where requests of service occur according to a Poisson distribution with intensity f_i. As customers arrive at a facility j from different demand nodes, the request for service at this facility is the sum of several Poisson processes with an intensity of λ_j, calculated as:

$$\lambda_j = \sum_{i \in N} f_i x_{ij} \tag{2.8}$$

2.4 Proposed ABC for PMCLAP

The proposed Artificial Bee Colony based solution for the foregoing PMCLAP is described in this section. Fig. 2.1 demonstrates the overall flow of execution of the proposed ABC-based solution of PMCLAP . And, in the following subsections, each step of the proposed procedure is described in detail.

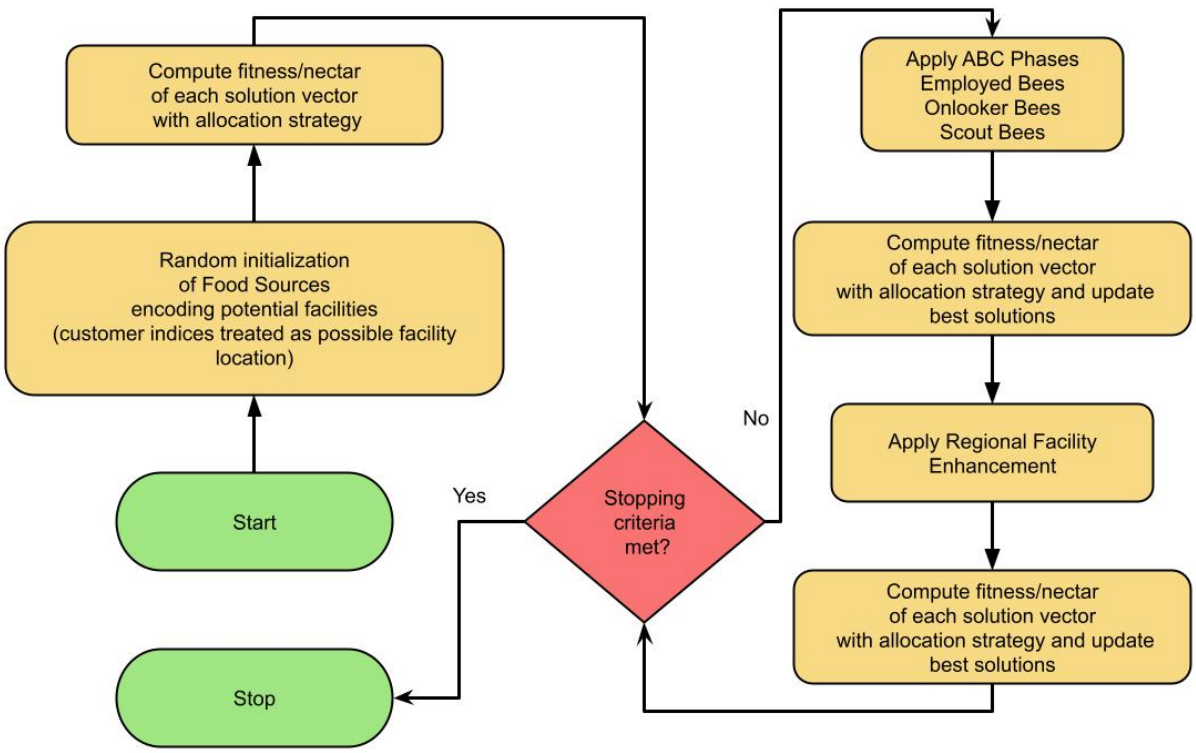

Figure 2.1: Flowchart demonstrating the proposed ABC-based procedure for solving PMCLAP

2.4.1 Solution encoding

In the ABC algorithm, solutions to the optimization problem are encoded as a string of indices (integer) in a similar way to the chromosome encoding in genetic algorithms Atta et al. [2018]. A set of potential candidate locations with k number of facilities is chosen from the set of m customers P = {p_1, p_2, ..., p_m} representing a possible solution with k facilities. Therefore, a food source encodes an integer string {t_1, t_2, ..., t_k} having length k representing the indices of the k customers selected as the facilities from the pool of customers. Each element, $p_i \in$ P for i={1, 2, ..., k}, since the potential k facilities locations are limited to the locations of the customers themselves.

2.4.2 Solution Vector (Food Sources) initialization

The initial colony of the bees comprises of P solutions where P is a user-defined parameter called solution vector or colony size. Selecting k random indices from the set {1,2,...,m} each food source of the initial colony is created. Here, we set P as 20, chosen experimentally.

2.4.3 Objective function computation

Objective function or nectar of a solution/food source represents the quality or goodness of the food source embedded within it with respect to PMCLAP. The objective of PMCLAP is to maximize the coverage (i.e., the total demands of the customers covered by some facilities satisfying at least the minimum service quality). Hence coverage of the solution encoded in a food source is considered as the objective function of the food source, as shown in Eq. 2.1. The allocation strategy of the PMCLAP ensures that the nectar collection from food sources is subject to the constraints of distance and congestion, thus avoiding invalid solutions that would violate these conditions. The objective function is to be maximized.

2.4.4 Allocation of customers

Multiple customers may be available within the service radius of a particular facility, and choosing the appropriate customers are crucial for achieving an optimal solution. The allocation strategy of the PMCLAP ensures that the nectar collection from food

sources is subject to the constraints of distance and congestion, thus avoiding invalid solutions that would violate these conditions. Several allocation strategies have been proposed by researchers, including hybrid approaches [de Assis Corrêa et al., 2007]. To achieve faster customer allocation, we employed three strategies interchangeably to obtain better experimental results: allocating the customer to the least congested feasible facility [de Assis Corrêa et al., 2009], allocating the customers with the maximum weighted demand to the facilities (proposed strategy), and allocating the customer to the random feasible facility. In the initial 50 epochs/iterations of the study, all three strategies for the computation of nectar in food sources are simultaneously employed, and the strategy yielding the highest nectar value is selected. Subsequently, in the remaining iterations/epochs until the termination criterion is satisfied, the computation of nectar follows a roulette selection approach based on the frequency of each strategy being chosen during the previous iterations. In our research investigation on the data-sets, we have observed that the time allocation strategy predominantly applied was focused on the least congested facility, accounting for 60 percent of the occurrences. Additionally, a random allocation approach for assigning feasible facilities to customers was utilized in 10 percent of the cases, while the remaining 30 percent of the instances involved allocation based on demand/distance factors. All the three strategies have been briefly shown in algorithms 6, 7, 8.

2.4.5 Swarm-phases in ABC Algorithm

Employed Bees

As Karaboga [2010] mentioned, employed bees looks for fresh food sources with more nectar within the neighbourhood of the food source in their colony. They find a neighbour fresh food source and then compute its quality of the nectar for which greedy strategy is used. For each food source in the colony, one random neighbour is chosen and if the chosen neighbour has more nectar then it is taken into food sources otherwise we go with the original food source.

Finding of fresh food source can be done with eq.2.9

$$\vec{Y_i} = \vec{X_i} + \phi(\vec{X_i} - \vec{X_k}) \tag{2.9}$$

Algorithm 6 GET-LEAST-CONGESTED-FACILITY

// Inputs: customer, solutionVector, congestionVector
// Outputs: facilityNumber, congestionVector, flag
// customer denotes the customer index we're trying to allocate a feasible facility
// solutionVector contains the indices of opened facilities
// congestionVector contains the congestion data of respective facilities so far accumulated for serving customers
// Intialize an empty vector availableFacility
// For constraint 5 For constraint 5 $x \leftarrow \mu \cdot ((1-\alpha)^{\frac{1}{b+2}})$
// For constraint 6 $x \leftarrow \mu + \left(\frac{\log(1-\alpha)}{\tau}\right)$
// K is the number of facilities opened

flag $\leftarrow$ false
$f \leftarrow 0.01 \cdot$ demand(customer)
min $\leftarrow -1$
facilityNumber $\leftarrow -1$
availableFacility $\leftarrow \emptyset$
for $i = 1$ to K **do**
    if distance(solution(i), customer) $\leq r$ **and** (congestionVector(i) + f) $\leq x$ **then**
        min $\leftarrow$ congestionVector(i)
        facilityNumber $\leftarrow i$
        availableFacility(i) $\leftarrow 1$
if min $\neq -1$ **then**
    for $i = 1$ to K **do**
        if availableFacility(i) $= 1$ **and** min $>$ congestionVector(i) **then**
            min $\leftarrow$ congestionVector(i)
            facilityNumber $\leftarrow i$
if facilityNumber $\neq -1$ **then**
    congestionVector(i) $\leftarrow$ congestionVector(i) + f
    flag $\leftarrow$ true

Algorithm 7 GET-MAX-WEIGHTED-CUSTOMER

// Inputs: facilityNumber, congestionVector
// Outputs: customer, congestionVector
// customer denotes the customer index we're trying to allocate to facilityNumber
// solutionVector contains the indices of opened facilities
// congestionVector contains the congestion data of respective facilities so far accumulated for serving customers
// Intialize an empty vector availableFacility
// For constraint 5 For constraint 5 $x \leftarrow \mu \cdot ((1-\alpha)^{\frac{1}{b+2}})$
// For constraint 6 $x \leftarrow \mu + \left(\frac{\log(1-\alpha)}{\tau}\right)$
// K is the number of facilities opened

$val \leftarrow 0$
$weightedMatrix \leftarrow \text{zeros}(1, m)$
for $i = 1$ to m **do**
    $f \leftarrow 0.01 \cdot \text{demand}(i)$
    congestionCheck $\leftarrow$ (congestionVector(facilityNo) + f) $\leq x$
    distanceCheck $\leftarrow$ distance(facilityNo, i) $\leq r$
    if notAllocated($customer$) **and** distanceCheck **and** congestionCheck **then**
        $weightedMatrix(1, i) \leftarrow \frac{\text{demand}(i)}{\text{distance(facilityNo},i)}$
customer $\leftarrow$ getMaxIndex($weightedMatrix$)
congestionVector $\leftarrow$ congestionVector(facilityNo) + f

Algorithm 8 GET-RANDOM-FACILITY

// Inputs: customer, solutionVector, congestionVector
// Outputs: facilityNumber, congestionVector, flag
// customer denotes the customer index we're trying to allocate a feasible facility
// solutionVector contains the indices of opened facilities
// congestionVector contains the congestion data of respective facilities so far accumulated for serving customers
// Intialize an empty vector availableFacility
// For constraint 5 For constraint 5 $x \leftarrow \mu \cdot ((1-\alpha)^{\frac{1}{b+2}})$
// For constraint 6 $x \leftarrow \mu + \left(\frac{\log(1-\alpha)}{\tau}\right)$
// K is the number of facilities opened

$flag \leftarrow$ false
$f \leftarrow 0.01 \cdot \text{demand}(\text{customer})$
$j \leftarrow 1$
for $i = 1$ to K **do**
    distanceCheck $\leftarrow$ distance(solution(i),customer) $\leq r$
    congestionCheck $\leftarrow$ (congestionVector(i) + f) $\leq x$
    if distanceCheck and congestionCheck **then**
        availableFacility(end+1) $\leftarrow i$
        $j \leftarrow j + 1$
        $flag \leftarrow$ true
if flag **then**
    randIndex $\leftarrow$ genRandomIndex(availableFacility)
    facilityNo $\leftarrow$ availableFacility(randIndex)
    y $\leftarrow$ congestionVector(facilityNo) + f
    congestionVector(facilityNo) $\leftarrow$ y

Here, $\vec{X}_i$ represents the ith food source and P is the colony size. The number of employed bees and the onlooker bees are equal to the colony size (P), k is an index randomly chosen from $\{1, \ldots, P\}$ and $k \neq i$.The coefficient ϕ is randomly generated integer from $[-1, 1]$. In case, the solution generated by eq. 2.9, $\vec{Y}_i$ is having less nectar than $\vec{X}_i$, an abandonment counter C_i is increased by one.

Onlooker Bees

Th onlooker bees of the Artificial Bee Colony Algorithm choose a food source based on the solutions supplied by employed bees, according to their probability [Karaboga, 2010]. This may be done with the eq. (2.10.

$$p_i = \frac{fit_{\mathrm{i}}}{\sum_{j=1}^{P} fit_{\mathrm{j}}} \tag{2.10}$$

, where p_i is the probability of i^{th} solution of the colony and fit_i denotes the nectar of solution i. Onlooker bees choose solutions $\vec{X}_i$ based on the probabilities of the solution through roulette wheel selection. And new solution $\vec{Y}_i$, within the neighbourhood of $\vec{X}_i$, can be generated using the eq. (2.9). Greedy strategy is applied between $\vec{X}_i$ and $\vec{Y}_i$ and richer source having higher nectar is chosen, leading to positive feedback behaviour. In case, the solution generated by eq. 2.9, $\vec{Y}_i$ is having less nectar than $\vec{X}_i$, an abandonment counter C_i is increased by one.

Scout Bees

The scouts in the ABC algorithm refer to the unemployed bees who chooses their food sources randomly after an abandonment criteria [Karaboga, 2010]. If an employed bee's solution cannot be improved through a predetermined number of trials, which is specified by the user and referred to as the "limit" or "abandonment criteria", here taken as L, the employed bee becomes a scout and abandons its solution. In this problem, we set L as $\lfloor 0.6 \cdot P \cdot k \rfloor$ similar to Atta et al. [2022], where P is the colony size and k is the number of facilities to be opened. The abandoned solution is then randomly searched by the scout to find a new solution. Therefore, poor sources, whether initially or due to exploitation, are abandoned, which creates a negative feedback behavior to balance the positive feedback.

Regional Facility Enhancement

After the scout bees phase, each facility in every food source (solution) of the colony is enhanced based on its region. This can be done with the proposed strategy mentioned at 9.

Algorithm 9 REGIONAL-FACILITY-ENHANCEMENT

```
// Inputs: Colony, P, k, N
// Output: Colony
neighbourSize ← round(N/10)
Updated_Colony ← Colony
for i ← 1 to P do
    for j ← 1 to k do
        Neighbours ← getClosestNeighbours(Colony(i, j), neighbourSize)
        candidateFacility ← randomly pick one neighbour from Neighbours
        Updated_Colony(i, j) ← candidateFacility
        if nectar(Updated_Colony(i, :)) ≥ nectar(Colony(i, :)) then
            Colony(i, :) ← Updated_Colony(i, :)
```

For each facility in a solution vector $\vec{X}_i$, it generates a vector $Neighbours$ containing its $N\%$ closest neighbours, and randomly picks one candidate facility from $Neighbours$. Then it replaces the original facility with the new candidate facility chosen in the solution vector $\vec{X}_i$. If the newly generated solution vector with the candidate facility has more nectar, it updates the solution vector with the new solution vector otherwise it keeps it unchanged. This enhancement procedure is then continued for remaining facilities of the updated solution vector. This strategy helps to find better solution vector and converge quickly.

Update best solutions

To prevent loss of promising solutions resulting from the stochastic nature of the ABC swarm phases, it is necessary to store the best solutions obtained up to the current generation/iteration. To achieve this, the updated colony currently in the memory is merged with the previous colony. Then a new colony is formed with best P solutions having higher nectar from the merged colony, and this is stored in the memory. This approach ensures that the best food source found thus far is retained. And this strategy is applied after the ABC phases and also after the enhancement procedure, just to ensure that quality solutions are not lost.

Stopping criterion

The iterative process of nectar computation involves employed bees, onlooker bees, scout bees, regional facility enhancement, and memorization of the best solutions across multiple generations. The iterative process stops if the best nectar value remains unchanged for the last hundred iterations. The output of ABC with regional facility enhancement is determined by the best quality solution, which corresponds to the highest total demand served.

2.5 Experimental Results

To asses the quality of the ABC with regional facility enhancement procedure, this section provides the experimental results and details. The algorithm was coded in Matlab™ version: R2021a. Computational experiments of the data-set used were done on machines equipped with an Intel i5™ processor running at 2.5 GHz frequency needing less than 500 Megabytes of RAM memory.

Similar to the approach in Pereira et al. [2015], benchmark data-sets of instances consisting of three types: 30 nodes, 324 nodes, and 818 nodes. Each instance was solved thirty times with the number of facilities ranging from two to fifty. For the instances with 30 nodes, the service radius (r) was set to 1.5 miles, for instances with 324 nodes r is set to 250 m, and for instances with 818 nodes r is set to 750 m. Marianov and Serra [1998] proposed the 30-node data-set, while the 324 and 818-node data-sets were introduced by de Assis Corrêa et al. [2007]. The instances can be found at `http://www.lac.inpe.br/~lorena/instancias.html`.

The tables 2.1, 2.2, and 2.3 provide the names of the instances along with the parameter values used for each instance. The name of an instance, such as 30_2_0_0_85, indicates that it incorporates a 30-node problem, where number of facilities opened is 2, the congestion type is based on the number of customers (0 for queue size, 1 for waiting time), the congestion parameter is either the number of clients b on the queue or the waiting time τ in minutes, and the minimum probability α is given as a percentage value. For the 30-node network, the rate parameter μ is fixed at 72, while for the 324 and 818 data-sets, it is fixed at 96. The parameter f_i that appears in formulations (2.1)-(2.7) is calculated as fd_i, where f is 0.01 for the 324- and 818-node networks, and either 0.015 (for queue size

Table 2.1: Experimental Results for 30-Node dataset using ABC with enhancement compared with others

Instance	CPLEX		MS Heuristic [12]		GRASP [6]			CS [4]			ANLS [14]			Proposed ABC with Enhancement				
	Best	Time	Best	Time	Best	Average	Time	Best	Average	Time	Best	Average	Time	Best	Average	Standard Deviation	Gap (%)	Time (s)
30_2_0_0_85	**3700**	0	**3700**	0.000	**3700**	3700	0.007	**3700**	3700	0.009	**3700**	3700	12.333	**3700**	3700	0	**0**	2.28
30_2_0_1_85	**5100**	0	4630	0.000	5090	5090	0.016	5090	5090	0.018	**5100**	5100	9.000	5090	5090	0	0.19	2.51
30_2_0_2_85	**5210**	0	4780	0.000	**5210**	5210	0.015	**5210**	5210	0.016	**5210**	5210	4.667	**5210**	5210	0	**0**	2.63
30_2_0_2_95	**4520**	0	4470	0.000	**4520**	4520	0.013	**4520**	4520	0.015	**4520**	4520	11.333	**4520**	4520	0	**0**	2.85
30_3_0_0_85	**5390**	0	5210	0.000	**5390**	5390	0.025	**5390**	5390	0.025	**5390**	5390	5.000	**5390**	5390	0	**0**	3.17
30_3_0_1_85	**5390**	0	5210	0.000	**5390**	5390	0.024	**5390**	5390	0.024	**5390**	5390	3.000	**5390**	5390	0	**0**	3.30
30_3_0_1_95	**5270**	6	5080	0.000	5240	5240	0.024	5240	5240	0.024	**5270**	5270	44.667	5260	5244	8	0.18	3.15
30_3_0_2_85	**5390**	1	5210	0.000	**5390**	5390	0.022	**5390**	5390	0.023	**5390**	5390	3.667	**5390**	5390	0	**0**	3.29
30_3_0_2_95	**5390**	0	5230	0.000	**5390**	5390	0.023	**5390**	5390	0.027	**5390**	5390	3.667	**5390**	5390	0	**0**	3.25
30_3_1_49_90	**2160**	0	**2160**	0.000	**2160**	2160	0.007	**2160**	2160	0.009	**2160**	2160	13.333	**2160**	2160	0	**0**	3.56
30_4_0_1_95	**5390**	0	5260	0.000	**5390**	5390	0.022	**5390**	5390	0.026	**5390**	5390	8.333	**5390**	5390	0	**0**	3.96
30_4_1_42_85	**4600**	0	4550	0.000	**4600**	4600	0.016	**4600**	4600	0.022	**4600**	4600	105.667	**4600**	4600	0	**0**	5.45
30_4_1_48_90	**1920**	0	1890	0.000	**1920**	1920	0.011	**1920**	1920	0.014	**1920**	1920	17.333	**1920**	1920	0	**0**	5.70
30_4_1_49_90	**2880**	0	2870	0.000	**2880**	2880	0.008	**2880**	2880	0.013	**2880**	2880	15.000	**2880**	2871	3	**0**	3.28
30_5_0_0_95	**5330**	13	5210	0.000	**5330**	5312.8	0.032	**5330**	5317.6	0.041	**5330**	5330	288.667	**5330**	5314	22.44	**0**	4.62
30_5_1_40_85	**3050**	93	2910	0.000	**3050**	3033.2	0.016	**3050**	3033.2	0.021	**3050**	3050	234.000	3000	3000	0	1.63	3.95
30_5_1_42_85	**5390**	1	5210	0.000	**5390**	5390	0.027	**5390**	5390	0.035	**5390**	5390	15.000	**5390**	5390	0	**0**	5.75
30_5_1_48_90	**2400**	1	2280	0.000	**2400**	2398.8	0.013	**2400**	2398.8	0.019	**2400**	2400	26.667	2390	2390	0	0.41	4.45
30_5_1_50_90	**4700**	1	4670	0.000	**4700**	4700	0.018	**4700**	4700	0.028	**4700**	4700	42.667	**4700**	4698	4	**0**	4.82
30_6_0_0_95	**5410**	1	5390	0.000	5390	5390	0.029	**5410**	5391	0.037	**5410**	5410	62.333	**5410**	5391.33	4.98	**0**	5.47
30_6_1_40_85	**3610**	10803	3480	0.000	**3610**	3606.8	0.021	**3610**	3608.8	0.031	**3610**	3610	514.000	**3610**	3604	8	**0**	5.83
30_6_1_41_85	**5330**	10802	5120	0.000	5270	5270.0	0.028	**5330**	5286.4	0.038	**5330**	5330	322.333	**5330**	5274.33	18.39	**0**	6.03
30_6_1_50_90	**5390**	1	5060	0.000	**5390**	5390	0.027	**5390**	5390.0	0.038	**5390**	5390	31.667	**5390**	5390	0	**0**	7
30_7_1_40_85	**4060**	1	3860	0.000	**4060**	4060	0.028	**4060**	4060	0.039	**4060**	4060	301.00	**4060**	4060	0	**0**	6.41
30_7_1_41_85	**5410**	1	5300	0.000	5390	5390	0.023	5390	5390	0.035	**5410**	5410	61.333	5390	5390	0	0.37	6.61
30_8_1_41_85	**5470**	0	5390	0.000	**5470**	5470	0.019	**5470**	5470	0.032	**5470**	5470	8.333	**5470**	5470	0	**0**	8.05
Average	4533.1	835.577	4389.6	0.000	4527.7	4526.2	0.02	4530.8	4527.1	0.025	4533.1	4533.1	83.269	4529.2	4526.24	2.28	0.10	4.5

type constraints) or 0.006 (for waiting time type constraints) for the 30-node network. It is important to use the same unit format for all the parameter values when dealing with the waiting time constraint. The parameter μ does not require any conversion, but τ needs to be adjusted to match the unit as it appears in minutes. This can be done by calculating $1440/\tau$ and substituting it for τ in the formulation (2.6) (24 hours or 1440 minutes of total distribution space).

Tables 2.1-2.3 display the best and average solutions (if available), computation time, and standard deviation for the ABC algorithm with local refinement for each instance. The study found that for the 30 node dataset, the strategy proposed by de Assis Corrêa et al. [2009] - allocating to the least congested facility - achieved better results with ABC almost reaching the benchmark. For the 324 and 818 node datasets, allocating according to the weighted demand strategy led to better results, although neither dataset met the benchmark. The Gap in % shows the difference in percentage between the obtained and benchmark results. Although not achieving the benchmark, the ABC algorithm had significantly faster convergence than ANLS Pereira et al. [2015].

The implementation of the proposed algorithm and all related data can be found here: `https://tinyurl.com/PMCLAP-using-ABC`

2.6 Conclusion

In conclusion, this study proposed an approach to solve the Probabilistic Maximal Coverage Location-Allocation Problem using the Artificial Bee Colony algorithm with regional facility enhancement strategy. Three allocation sub-problems were solved using different strategies, resulting in promising results for the 30 node data-set and 818 node data-set, in terms of both achieving benchmark results and computational time. It achieves the optimal benchmark results of CPLEX in 50% of time, with an average computational time of 85.83 seconds. In comparison to non-CPLEX strategies like for the instances shown in table 2.1-2.3: MS Heuristics which achieves the benchmark results only 2.70% of the time; GRASP achieves the benchmark results 29.72% time, and CS achieves the benchmark results 58.10% time. The heuristic method finds optimal solutions for 21 out of 26 instances of the 30-node data-set with an average gap of 0.10%, for none of the 24 instances of the 324-node data-set with an average gap of 0.17%, and for 16 out of 24

Table 2.2: Experimental Results for 324-Node dataset using ABC with enhancement compared with others

Instance	CPLEX		MS Heuristic [12]		GRASP [6]			CS [4]			ANLS [14]			Proposed ABC with Enhancement				
	Best	Time	Best	Time	Best	Average	Time	Best	Average	Time	Best	Average	Time	Best	Average	Standard Deviation	Gap (%)	Time (s)
324_10_0_0_85	**37180**	13	37081	0.220	37157	37155.8	2.24	37173	37163.2	3.460	**37180**	37180	1255.333	37172	37161.60	5.49	0.02	38.28
324_10_0_0_95	**21460**	9	21386	0.140	21446	21441.2	2.05	21455	21447.2	3.160	**21460**	21460	865.667	21456	21450.60	3.97	0.01	46.60
324_10_0_1_85	**51000**	22	50750	0.200	50948	50946.3	2.30	50948	50946.3	3.410	**51000**	51000	1435.000	50944	50919.50	22.57	0.10	44.64
324_10_0_1_95	**35360**	14	35250	0.160	35339	35336.5	2.06	35359	35354.6	3.170	**35360**	35360	857.000	35357	35355.90	1.70	0.08	36.35
324_10_0_2_85	**59740**	22	59598	0.200	59693	59688.5	2.24	59693	59688.5	3.330	**59740**	59740	987.667	59666	59644.50	12.31	0.12	35.80
324_10_0_2_95	**45390**	9	45300	0.200	45341	45334.6	2.01	45374	45354.6	3.100	**45390**	45390	1292.000	45366	45359.80	3.31	0.05	49.25
324_10_1_40_85	**27700**	14	27283	0.170	27670	27666.6	2.18	27698	27692.1	3.370	**27700**	27700	1406.333	27692	27687.30	3.95	0.02	38.51
324_10_1_41_85	**29360**	6	29288	0.140	29658	29656.1	2.24	29351	29341.9	3.520	**29360**	29360	1182.667	29350	29346.60	2.57	0.03	37.78
324_10_1_42_85	**30950**	11	30902	0.170	30928	30920.8	2.21	30948	30943.3	3.330	**30950**	30950	1034.667	30947	30944.10	3.20	0.009	39.62
324_10_1_48_90	**26920**	15	26885	0.140	26899	26896.6	2.18	26917	26910.6	3.500	**26920**	26920	1492.667	26914	26907.10	4.34	0.02	49.02
324_10_1_49_90	**28330**	22	28206	0.250	28295	28285.0	2.24	28318	28310.3	3.430	**28330**	28330	1747.000	28317	28310.80	4.21	0.01	36.58
324_10_1_50_90	**29680**	6	29638	0.220	29658	29656.1	2.24	29672	29665.5	3.360	**29680**	29680.0	1127.33	29669	29665.30	2.75	0.03	48.54
324_20_0_0_85	**74360**	198	73981	0.830	74165	74106.0	4.80	74165	74106.7	9.710	**74360**	74357.7	6191.667	74188	74039.10	89.77	0.23	96.91
324_20_0_0_95	**42920**	22	42714	0.730	42813	42792.2	4.24	42840	42804.9	9.370	**42920**	42919.7	3936.333	42852	42835.50	12.04	0.15	77.06
324_20_0_1_85	**102000**	612	100628	1.020	101374	101177.4	4.88	101374	101177.4	9.070	101978	101974	7212.667	101045	100600.10	273.21	0.93	105.92
324_20_0_1_95	**70720**	55	70368	0.740	70561	70529.8	4.43	70656	70628.2	9.060	**70720**	70720	4417.333	70762	70641.20	11.26	0.05	102.59
324_20_0_2_85	**119740**	10804	118451	0.770	118771	118613.6	4.86	118771	118613.6	8.990	119455	119447.3	6388.000	118193	117927.40	169.44	1.29	97.45
324_20_0_2_95	**90780**	74	90424	0.810	90550	90518.9	4.64	90556	90521.2	9.400	**90780**	90780	6667.333	90604	90529.90	50.54	0.19	68.40
324_20_1_40_85	**55400**	52	55006	0.840	55193	55147.4	4.63	55306	55226.7	9.650	55396	55398.3	5488.667	55303	55249.50	41.96	0.17	72.03
324_20_1_41_85	**58720**	85	58577	1.020	58602	58582.6	4.72	58604	58583.4	10.330	**58720**	58720	7417.667	58626	58515.40	81.26	0.16	109.16
324_20_1_42_85	**61900**	35	61637	0.880	61746	61715.1	4.68	61847	61822.6	9.740	**61900**	61900	3561.333	61849	61831.30	14.10	0.01	102.38
324_20_1_48_90	**53840**	49	53377	0.630	53647	53607.5	4.54	53793	53689.5	9.820	53839	53838.7	5912.667	53760	53711.90	36.52	0.14	88.30
324_20_1_49_90	**56660**	167	56180	1.340	56321	56254.3	4.59	56532	56429.9	9.550	**56660**	56660	7208.667	56498	56382.40	84.09	0.28	108.13
324_20_1_50_90	**59360**	56	59119	0.940	59253	59237.0	4.74	59285	59242.6	10.080	**59360**	59360	5551.333	59287	59262.50	18.91	0.12	102.29
Average	52883.3	515.5	52595.8	0.532	52751.1	52719.4	3.41	52776.5	52736.0	6.455	52881.5	52880.9	3523.292	52743.8	52681.8	38.8	0.17	67.14

Table 2.3: Experimental Results for 818-Node dataset using ABC with enhancement compared with others

Instance	CPLEX		MS Heuristic [12]		GRASP [6]			CS [4]			ANLS [14]			Proposed ABC with Enhancement				
	Best	Time	Best	Time	Best	Average	Time	Best	Average	Time	Best	Average	Time	Best	Average	Standard Deviation	Gap (%)	Time (s)
818_10_0_0_95	**21460**	557	21429	4.94	21459	21458.3	6.30	**21460**	21459.9	11.23	**21460**	21460	6818	**21460**	21460	0	**0**	60.37
818_10_0_1_95	**35360**	657	35339	13.05	**35360**	35360	6.65	**35360**	35360	11.54	**35360**	35360	10407	**35360**	35359	1.54	**0**	61.82
818_10_0_2_95	**45390**	648	45375	12.09	45389	45388.2	6.97	**45390**	45390	12.17	**45390**	45390	7400	**45390**	45389.40	0.80	**0**	62.96
818_10_1_48_90	**26290**	586	26907	11.38	26899	26896.6	2.18	**26920**	26920	11.32	**26920**	26920	8212	**26920**	26919.60	1.20	**0**	103.70
818_10_1_49_90	**28330**	659	28309	8.63	28295	28285	2.24	**28330**	28330	12.23	**28330**	28330	8052.6667	**28330**	28328.10	2.25	**0**	83.17
818_10_1_50_90	**29680**	682	29661	18.28	29658	29656.1	2.24	**29680**	29680	11.33	**29680**	29680	9078.333	**29680**	29679.90	0.30	**0**	83.07
818_20_0_0_85	**74360**	785	74313	75.05	74353	74352.1	16.27	**74360**	74359.8	66.81	**74360**	74360	17183.667	**74360**	74355	3.92	**0**	102.90
818_20_0_0_95	**42920**	562	42793	25.67	42903	42900	14.51	**42920**	42918.9	54.84	**42920**	42920	9890.333	42918	42917.10	0.53	0.004	96.32
818_20_0_1_85	**102000**	1491	101933	76.06	101996	101991.9	19.24	**102000**	101998.9	67.36	**102000**	102000	16602	**102000**	101990.80	10.21	**0**	103.511
818_20_0_1_95	**70720**	610	70644	37.03	70717	70715.1	16.90	**70720**	70719.2	62.91	**70720**	70720	16880.667	**70720**	70717.80	1.77	**0**	96.18
818_20_0_2_85	**119480**	1579	119397	105.48	119468	119466	21.26	**119480**	119477.6	74.36	**119480**	119480	19074.667	**119480**	119472.20	13.15	**0**	104.08
818_20_0_2_95	**90780**	841	90730	75.19	90772	90769.5	19.18	**90780**	90779.6	66.80	**90780**	90780	14622.667	**90780**	90774.20	4.48	**0**	102.11
818_20_1_40_85	**55400**	833	55325	66.11	55193	55147.4	4.63	**55400**	55399	61.04	**55400**	55400	15044.333	**55400**	55395.30	8.92	**0**	163.10
818_20_1_41_85	**58720**	641	58637	69.63	NA	NA	NA	**58720**	58719.9	57.31	**58720**	58720	14384	**58720**	58715.90	7.35	**0**	169.17
818_20_1_42_85	**61900**	929	61814	71.81	NA	NA	NA	**61900**	61898.8	58.71	**61900**	61900	14487	61899	61886	17.64	0.001	157.11
818_20_1_48_90	**53840**	640	56602	34.34	NA	NA	NA	**53840**	53839.6	59.56	**53840**	53840	13808.667	**53840**	53836.30	4.51	**0**	155.97
818_20_1_49_90	**56660**	948	56602	34.34	NA	NA	NA	**56660**	56660	67.42	**56660**	56660	13008.333	**56660**	56655.10	4.84	**0**	166.31
818_20_1_50_90	**59360**	877	59269	39.84	NA	NA	NA	**59360**	59359.9	58.48	**59360**	59360	14568.667	**59360**	59357	3.76	**0**	161.33
818_50_0_0_85	**185900**	2956	185426	756.72	1857791	185755.8	50.62	185880	185775.6	364.20	185637	185587.7	24094.33	185855	185801	59.16	0.02	339.63
818_50_0_1_85	**255000**	6332	254509	730.20	254860	254836.6	58.37	254985	254905.0	387.37	254996	254987.3	23978	254934	254868.50	61.11	0.02	360.95
818_50_0_2_85	**298700**	4435	298217	757.73	2988547	298511.2	63.58	298582	298517.6	392.29	298692	298648.3	23978	298548	298344.50	114.98	0.05	390.01
818_50_1_48_90	**134600**	785	134088	596.11	NA	NA	NA	134598	134561.6	335.000	134597	134593.3	24094.333	134592	134578.70	10.24	0.05	481.34
818_50_1_49_90	**141650**	1719	141281	571.17	NA	NA	NA	141586	141532.8	356.800	**141650**	141606.7	24425	141614	141597	13.09	0.02	506.78
818_50_1_50_90	**148400**	1946	147931	667.67	NA	NA	NA	148383	148321.9	337.400	148397	148378	25031.667	148366	148327.30	26.88	0.02	510.95
Average	91563.8	1363.417	91402.4	207.934	NA	NA	NA	91605.3	91598.9	206.497	91631.6	91625.5	13801.309	91609.8	91595.0	11.979	0.008	144.127

instances of the 818-node data-set with an average gap of 0.007%. Overall, the heuristic method attains 50% of the optimal solutions achieved by CPLEX, with an average gap of 0.09% for the standard instances tested. The quality of the heuristic solutions depends largely on the customer allocation strategy. Future research will focus on developing improved allocation strategies to enhance the performance of the heuristic method.

Bibliography

S. Atta, P. R. Sinha Mahapatra, and A. Mukhopadhyay. Solving maximal covering location problem using genetic algorithm with local refinement. *Soft Computing*, 22:3891–3906, 2018.

S. Atta, P. R. S. Mahapatra, and A. Mukhopadhyay. Solving a new variant of the capacitated maximal covering location problem with fuzzy coverage area using metaheuristic approaches. *Computers & Industrial Engineering*, 170:108315, 2022. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2022.108315. URL https://www.sciencedirect.com/science/article/pii/S0360835222003710.

R. Church and C. ReVelle. The maximal covering location problem. In *Papers of the regional science association*, volume 32, pages 101–118. Springer-Verlag Berlin/Heidelberg, 1974.

F. de Assis Corrêa, A. A. Chaves, and L. A. N. Lorena. Hybrid heuristics for the probabilistic maximal covering location-allocation problem. *Operational Research*, 7: 323–343, 2007.

F. de Assis Corrêa, L. A. N. Lorena, and G. M. Ribeiro. A decomposition approach for the probabilistic maximal covering location-allocation problem. *Computers & Operations Research*, 36(10):2729–2739, 2009.

T. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

Y.-Y. Huang, Q.-K. Pan, L. Gao, Z.-H. Miao, and C. Peng. A two-phase evolutionary algorithm for multi-objective distributed assembly permutation flowshop scheduling problem. *Swarm and Evolutionary Computation*, 74:101128, 2022.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

D. Karaboga. Artificial bee colony algorithm. *scholarpedia*, 5(3):6915, 2010.

D. Karaboga et al. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer . . . , 2005.

V. Marianov and D. Serra. Probabilistic, maximal covering location—allocation models forcongested systems. *Journal of Regional Science*, 38(3):401–424, 1998.

A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay. A survey of multiobjective evolutionary clustering. *ACM Computing Surveys (CSUR)*, 47(4):1–46, 2015.

M. A. Pereira, L. C. Coelho, L. A. Lorena, and L. C. De Souza. A hybrid method for the probabilistic maximal covering location–allocation problem. *Computers & Operations Research*, 57:51–59, 2015.