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

Subhrajit Das
Supervisors:
Dr. Priya Ranjan Sinha Mahapatra
Dr. Soumen Atta
Department of Computer Science and Engineering
University of Kalyani
iamsubhrajit10@gmail.com

June, 2023

***Abstract*—This report proposes a solution to the NP-Hard Probabilistic Maximal Covering Location-Allocation Problem (PMCLAP) using the Artificial Bee Colony algorithm with Regional Facility Enhancement. The approach is aimed at improving the results by applying the regional facility enhancement procedure for enhanced results and quick convergence. The PMCLAP problem involves another sub-problem, the allocation of specific customers to appropriate facilities to attain optimality, and for that several strategies are suggested. The effectiveness of the proposed solving strategy is discussed, and its capability in addressing complex PMCLAPs is highlighted. The approach has been tested on three data-sets of different sizes: 30 instances, 324 instances, and 818 instances. The result achieved for the 30 & 818-Node data-set looks promising while for the 324-Node data-set a minor gap is noticed compared to the results achieved by CPLEX solver.**

KEYWORDS

Facility location-allocation problem, Covering location allocation problem, Probabilistic maximal covering location-allocation problem (PMCLAP), Artificial bee colony algorithm (ABC), Regional facility enhancement

CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. Introduction

Introduced by Marianov and Serra [1998], the probabilistic maximal covering location-allocation problem (PMCLAP) is a renowned 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 information gathered from the previous iterations. 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 1 shows the basic process of ABC Algorithm Karaboga [2010]

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 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 total 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, MS Heuristic, CS, GRASP, CPLEX 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 report is arranged as follows: Section II discusses the similar works; PMCLAP is mathematically defined in section III; Section IV describes the suggested ABC technique in detail; the experimental results are reported and discussed in the Section V; and at the end report is

Algorithm 1 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 (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. (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 $\mathbf{X}_g$

concluded in Section VI .

II. Literature Review

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 minimum quality on the service level, 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 proposed various methods 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. A particle swarm-optimization (PSO) algorithm was introduced by Huang et al. [2022] for solving the PMCLAP. The proposed method uses a pool of particles for finding the optimal solution. Based on the best solution found so far it updates the velocity and the position of the particles.

III. Problem definition

A graph consisting of n nodes in set N is considered for the problem definition where each node is assigned with a demand d_i and in case a facility is located at the facility candidate i, a service radius (or covering distance) S_i is also considered. 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 formulation of this problem can be as follows:

$$maximize \sum_{i \in N} \sum_{j \in N_i} d_i x_{ij} \tag{1}$$

subject to

$$\sum_{j \in N_i} x_{ij} \leq 1, \quad i \in N \tag{2}$$

$$\sum_{i \in N} y_j = p \tag{3}$$

$$x_{ij} \leq y_j, \quad i \in N, \ j \in N \tag{4}$$

$$\sum_{i \in N} f_i x_{ij} \leq \mu \sqrt[b+2]{1-\alpha} \quad j \in N \tag{5}$$

$$\sum_{i \in N} f_i x_{ij} \leq \mu + \frac{1}{\tau} \ln(1-\alpha), \quad j \in N \tag{6}$$

$$y_j, x_{ij} \in \{0,1\}, i \in N, j \in N \tag{7}$$

The optimization problem aims to maximize the total demand served, with constraints ensuring the allocation and location of facilities. Specifically, the objective function (1) maximizes the total demand/population served, while constraints (2) guarantee that at most one facility serves each client, and constraint (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 (4). Constraint (5) ensures that facility j has fewer than b clients/customers in the waiting queue with at least probability α, while constraint (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 (7). It is worth noting

that x_{ij} is set to zero for all $j \notin N_i$.

Constraints (5) and (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:

$$\lambda_j = \sum_{i \in N} f_i x_{ij} \tag{8}$$

IV. Proposed ABC for PMCLAP

The proposed Artificial Bee Colony based solution for the foregoing PMCLAP is described in this section. Fig. 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.

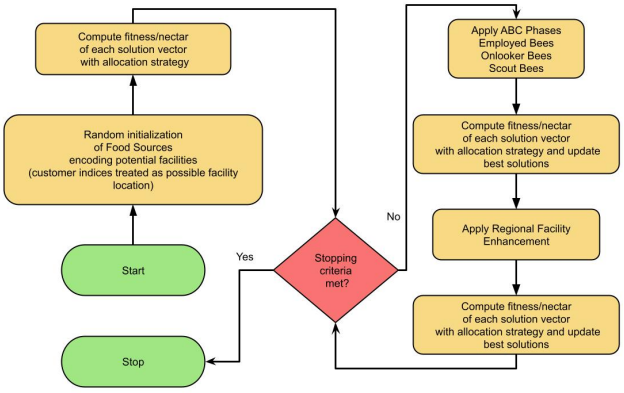

Fig. 1: Flowchart demonstrating the proposed ABC-based procedure for solving PMCLAP

A. 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, $p_{ti} \in$ P for i=$\{1, 2, ..., k\}$, since the potential k facilities locations are limited to the locations of the customers themselves.

B. 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=20, chosen experimentally.

C. 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. 1. The objective function is to be maximized.

D. Allocation of customers

Multiple facilities may be available within the service radius of a particular customer, and choosing the appropriate candidate facility is crucial for achieving an optimal solution. 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 customer according to the feasible facility with the maximum weighted demand, 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 datasets, 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 2, 3, 4, .

Algorithm 2 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

E. Swarm-phases in ABC Algorithm

1) 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.9

$$\vec{Y_i} = \vec{X_i} + \phi(\vec{X_i} - \vec{X_k}) \tag{9}$$

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. 9, $\vec{Y_i}$ is having less

Algorithm 3 GET-MAX-WEIGHTED-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)
$j \leftarrow 1$
$weightedMatrix \leftarrow zeros(1, K)$
// 1xK matrix initialized with zeros
for $i = 1$ to K **do**
    d $\leftarrow$ distance(solution(i),customer)
    distanceCheck $\leftarrow$ d $\leq r$ and d $\neq 0$
    congestionCheck $\leftarrow$ (congestionVector(i) $+ f) \leq x$
    if distanceCheck and congestionCheck **then**
        weightedMatrix(i) $\leftarrow$ demand(customer)/d
[maxWeight, i] $\leftarrow$ max(weightedMatrix)
facilityNo $\leftarrow$ i
if maxW $\neq 0$ **then**
    $flag \leftarrow$ true
    y $\leftarrow$ congestionVector(facilityNo) $+ f$
    congestionVector(facilityNo) $\leftarrow$ y

Algorithm 4 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
$f \leftarrow 0.01 \cdot$ demand(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

nectar than $\vec{X}_i$, an abandonment counter C_i is increased by one.

2) 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 probabilityKaraboga [2010]. This may be done with the eq. (10.

$$p_i = \frac{fit_{\mathrm{i}}}{\sum_{j=1}^{P} fit_{\mathrm{j}}} \tag{10}$$

, where p_i is the probability of i$^{\mathrm{th}}$ 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. (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. 9, $\vec{Y}_i$ is having less nectar than $\vec{X}_i$, an abandonment counter C_i is increased by one.

3) 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 $\lfloor 0.6 \cdot P \cdot k \rfloor$, 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.

4) 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 5.

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.

Algorithm 5 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, :)
```

5) 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.

6) 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.

V. 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

TABLE I: Experimental Results for 30-Node data-set using ABC with enhancement compared with others

Instance	ANLS			MS_Heuristic		CS			GRASP			CPLEX		ABC WITH Enhancement				
	Best	Average	Time	Best	Time	Best	Average	Time	Best	Average	Time	Best	Time	Best	Average	Standard Deviation	Gap (%)	Time(s)
30_2_0_0_85	3700	3700	12.333	3700	0.000	3700	3700	0.009	3700	3700	0.007	3700	0	3700	3700	0	0	2.28
30_2_0_1_85	5100	5100	9.000	4630	0.000	5090	5090	0.018	5090	5090	0.016	5100	0	5090	5090	0	0.19	2.51
30_2_0_2_85	5210	5210	4.667	4780	0.000	5210	5210	0.016	5210	5210	0.015	5210	0	5210	5210	0	0	2.63
30_2_0_2_95	4520	4520	11.333	4470	0.000	4520	4520	0.015	4520	4520	0.013	4520	0	4520	4520	0	0	2.85
30_3_0_0_85	5390	5390	5.000	5210	0.000	5390	5390	0.025	5390	5390	0.025	5390	0	5390	5390	0	0	3.17
30_3_0_1_85	5390	5390	3.000	5210	0.000	5390	5390	0.024	5390	5390	0.024	5390	0	5390	5390	0	0	3.30
30_3_0_1_95	5270	5270	44.667	5080	0.000	5240	5240	0.024	5240	5240	0.024	5270	6	5260	5244	8	0.18	3.15
30_3_0_2_85	5390	5390	3.667	5210	0.000	5390	5390	0.023	5390	5390	0.022	5390	1	5390	5390	0	0	3.29
30_3_0_2_95	5390	5390	3.667	5230	0.000	5390	5390	0.027	5390	5390	0.023	5390	0	5390	5390	0	0	3.25
30_4_0_1_95	5390	5390	8.333	5260	0.000	5390	5390	0.026	5390	5390	0.022	5390	0	5390	5390	0	0	3.96
30_5_0_0_95	5330	5330	288.667	5210	0.000	5330	5317.6	0.041	5330	5312.8	0.032	5330	13	5330	5314	22.44	0	4.62
30_6_0_0_95	5410	5410	62.333	5390	0.000	5410	5391	0.037	5390	5390	0.029	5410	1	5410	5391.33	4.98	0	5.47

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 tables I, II, and III 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 (1)-(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 type constraints) or 0.006 (for waiting time type constraints) for the 30-node network.

Tables I-III 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].

TABLE II: Experimental Results for 324-Node data-set using ABC with enhancement compared with others

Instance	ANLS			MS_Heuristic		CS			GRASP			CPLEX		ABC WITH Enhancement				
	Best	Average	Time	Best	Time	Best	Average	Time	Best	Average	Time	Best	Time	Best	Average	Standard Deviation	Current Gap (%)	Time(s)
324_10_0_0_85	37180	37180	1255.333	37081	0.220	37173	37163.2	3.460	37157	37155.8	2.24	37180	13	37172	37161.60	5.49	0.02	38.28
324_10_0_0_95	21460	21460	865.667	21386	0.140	21455	21447.2	3.160	21446	21441.2	2.05	21460	9	21456	21450.60	3.97	0.01	46.60
324_10_0_1_85	51000	51000	1435.000	50750	0.200	50948	50946.3	3.410	50948	50946.3	2.30	51000	22	50944	50919.50	22.57	0.10	44.64
324_10_0_1_95	35360	35360	857.000	35250	0.160	35359	35354.6	3.170	35339	35336.5	2.06	35360	14	35357	35355.90	1.70	0.08	36.35
324_10_0_2_85	59740	59740	987.667	59598	0.200	59693	59688.5	3.330	59693	59688.5	2.24	59740	22	59666	59644.50	12.31	0.12	35.80
324_20_0_0_85	74360	74357.7	6191.667	73981	0.830	74165	74106.7	9.710	74165	74106.0	4.80	74360	198	74188	74039.10	89.77	0.23	96.91
324_20_0_0_95	42920	42919.7	3936.333	42714	0.730	42840	42804.9	9.370	42813	42792.2	4.24	42920	22	42852	42835.50	12.04	0.15	77.06
324_20_0_1_85	101978	101974	7212.667	100628	1.020	101374	101177.4	9.070	101374	101177.4	4.88	102000	612	101045	100600.10	273.21	0.93	105.92
324_20_0_1_95	70720	70720	4417.333	70368	0.740	70656	70628.2	9.060	70561	70529.8	4.43	70720	55	70762	70641.20	11.26	0.05	102.59
324_20_0_2_85	119455	119447.3	6388.000	118451	0.770	118771	118613.6	8.990	118771	118613.6	4.86	119740	10804	118193	117927.40	169.44	1.29	97.45
324_20_0_2_95	90780	90780	6667.333	90424	0.810	90556	90521.2	9.400	90550	90518.9	4.64	90780	74	90604	90529.90	50.54	0.19	68.40

TABLE III: Experimental Results for 818-Node data-set using ABC with enhancement compared with others

Instance	ANLS			MS_Heuristic		CS			GRASP			CPLEX		ABC WITH Enhancement				
	Best	Average	Time	Best	Time	Best	Average	Time	Best	Average	Time	Best	Time	Best	Average	Standard Deviation	Current Gap (%)	Time(s)
818_10_0_0_95	21460	21460	6818	21429	4.94	21460	21459.9	11.23	21459	21458.3	6.30	21460	557	21460	21460	0	0	60.37
818_10_0_1_95	35360	35360	10407	35339	13.05	35360	35360	11.54	35360	35360	6.65	35360	657	35360	35359	1.54	0	61.82
818_10_0_2_95	45390	45390	7400	45375	12.09	45390	45390	12.17	45389	45388.2	6.97	45390	648	45390	45389.40	0.80	0	62.96
818_20_0_0_85	74360	74360	17183.67	74313	75.05	74360	74359.8	66.81	74353	74352.1	16.27	74360	785	74360	74355	3.92	0	102.90
818_20_0_0_95	42920	42920	9890.33	42793	25.67	42920	42918.9	54.84	42903	42900	14.51	42920	562	42918	42917.10	0.53	0.004	96.32
818_20_0_1_85	102000	102000	16602	101933	76.06	102000	101998.9	67.36	101996	101991.9	19.24	102000	1491	102000	101990.80	10.21	0	103.511
818_20_0_1_95	70720	70720	16880.67	70644	37.03	70720	70719.2	62.91	70717	70715.1	16.90	70720	610	70720	70717.80	1.77	0	96.18
818_20_0_2_85	119480	119480	19074.67	119397	105.48	119480	119477.6	74.36	119468	119466	21.26	119480	1579	119480	119472.20	13.15	0	104.08
818_20_0_2_95	90780	90780	14622.67	90730	75.19	90780	90779.6	66.80	90772	90769.5	19.18	90780	841	90780	90774.20	4.48	0	102.11
818_50_0_0_85	185637	185587.7	24094.33	185426	756.72	185880	185775.6	364.20	1857791	185755.8	50.62	185900	2956	185822	185801	59.16	0.04	339.63
818_50_0_1_85	254996	254987.3	23978	254509	730.20	254985	254905.0	387.37	254860	254836.6	58.37	255000	6332	254892	254868.50	61.11	0.04	360.95
818_50_0_2_85	298692	298648.3	23978	298217	757.73	298582	298517.6	392.29	2988547	298511.2	63.58	298700	4435	298263	298344.50	114.98	0.14	390.01

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisors, Dr. Priya Ranjan Sinha Mahapatra from the Department of Computer Science and Engineering, University of Kalyani, and Dr. Soumen Atta from the School of Engineering and Management,University of Nova Gorica for their guidance, support, and encouragement throughout my research. Their insights and expertise have been invaluable in shaping my work and pushing me to new heights.

I would also like to thank the Department of Computer Science and Engineering at the University of Kalyani for providing access to the lab resources that were crucial to the success of this project. The moral support and encouragement from the faculty and staff in the department have been a constant source of inspiration.

Finally, I would like to express my deep appreciation to my family and friends who have supported me throughout my academic journey. Their love, encouragement, and unwavering support have been the foundation of my success.

VI. 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. However, for the data-set 324, the benchmark results were missed by a small margin compared to the CPLEX result, however it beats some of the existing models like MS Heuristics, and also beats other models in terms of computational time by a multi-fold times. Future work will focus on making appropriate changes to achieve benchmark results for some instances while maintaining faster computational time. Overall, this study provides insights into solving the PMCLAP problem and opens up avenues for further research in this area.

REFERENCES

Soumen Atta, Priya Ranjan Sinha Mahapatra, and Anirban Mukhopadhyay. Solving maximal covering location problem using genetic algorithm with local refinement. *Soft Computing*, 22:3891–3906, 2018.

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

Francisco de Assis Corrêa, Antonio Augusto Chaves, and Luiz Antonio Nogueira Lorena. Hybrid heuristics for the probabilistic maximal covering location-allocation problem. *Operational Research*, 7:323–343, 2007.

Francisco de Assis Corrêa, Luiz Antonio Nogueira Lorena, and Glaydston Mattos Ribeiro. A decomposition approach for the probabilistic maximal covering location-allocation problem. *Computers & Operations Research*, 36(10):2729–2739, 2009.

Ying-Ying Huang, Quan-Ke Pan, Liang Gao, Zhong-Hua Miao, and Chen Peng. A two-phase evolutionary algorithm for multi-objective distributed assembly permutation flowshop scheduling problem. *Swarm and Evolutionary Computation*, 74:101128, 2022.

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

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

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

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