CHAPTER 3

MATERNAL HEALTHCARE FACILITY PLANNING WITH PENALTY ON OVERBURDENING

3.1 Introduction

In the previous chapter (Chapter 2), the planning framework did not allow for over allocation of MTBs to the health facilities. The fact is, particularly in Indian context, that health facilities are overburdened. This overburdening on one hand is because of large demand coming from huge population and the lack of funds for creating sufficient number of facilities. Under such a scenario, overburdening is unavoidable but is definitely undesirable. To bring a right balance between the two contradictory emerging scenarios, overburdening should be allowed but only at a cost. This cost in further elaborations and discussions is termed as penalty cost. The problem thus considered in this chapter is the same except for possibility of overburdening at a penalty cost.

As discussed in Chapter 2, the complexity of the HCFL model grows with the problem size, and the problem may become intractable. In fact, in countries such as India, such planning problems are large scale problems owing to many administrative regions consisting of more than a thousand villages. For example, as mentioned in earlier chapters, the Varanasi district of Uttar Pradesh state in India consists of 1333 villages. Seoni district of Madhya Pradesh state in India consists of 1593 villages, and the Kolar district of Karnataka state in India consists of 1809 villages (India Village Directory, 2021). There are many districts in India having more than a thousand villages. In these districts/ regions, some of the villages have a very small population and are at a short distance from the nearby populated village. During the planning of the network, planners can merge such villages to constitute a single location. Even after

merging, the total number of locations comes out in thousands. To plan a maternal healthcare network in such a large region using MILP formulation is a challenge for healthcare planners, particularly when one looks for a high-quality solution in a reasonable amount of time for a large-scale real-world problem. To solve the large size MILP of HFLPs, metaheuristics such as iterative local search (Brito et al., 2010), neighbourhood search (Hansen et al., 2010), simulated annealing (Kirkpatrick et al., 1983), and Tabu search (Glover and Laguna, 1998), etc., can be used population-based metaheuristics inspired by nature have also received a lot of attention in recent years. These metaheuristics are typically driven by evolution or swarm intelligence. Swarm intelligence-based algorithms, in particular, have drawn the attention of researchers due to their capability of exploration and exploitation. Following a thorough review of the literature, it appears that several swarm intelligence-based metaheuristics, such as particle swarm optimization (Li et al., 2021) and artificial bee colony (Akay et al., 2021), are extremely effective in solving diverse combinatorial optimization problems.

Out of all these approaches, particle swarm optimization (PSO), artificial bee colony (ABC), and the JAYA algorithm (Rao et al., 2016) have been used in this chapter to solve the considered maternal healthcare facility location-allocation problem. Extensive computational experiments have also been performed to analyze the comparative efficiency and usefulness of these metaheuristics. Besides, the impact of consideration of penalty cost is also analyzed on the resulting solution.

In this chapter, the issue of service quality due to stochastic demand is also addressed. The formulation considered in Chapter 2 assumes the service demand from MTBs to be deterministic. However, in reality, the volume handled by a healthcare facility from a location need not be constant but random. A heavy load on a day is bound to have a negative impact on

the quality of healthcare service. Thus, the Monte Carlo simulation approach is used to find how the service quality is going to be impacted by the amount of variability in demand originating from a location.

This chapter is organized as follows. The problem is detailed in Section 3.2, and the corresponding mathematical model in Section 3.3. The metaheuristics as solution approaches are described in Section 3.4. The results of computational experiments, along with related findings, are given in Section 3.5. The impact of change in demand and penalty is discussed in Section 3.6. The optimization and simulation framework is discussed in Section 3.7. Finally, Section 3.8 concludes the chapter.

3.2 The Problem

In case the capacity restrictions are removed or relaxed, the mathematical model of Chapter 2 will try to establish fewer number of facilities only to ensure that the required facilities are within the reach of MTBs. However, in this process, allocation of MTBs to a facility may be much more than the capacity. These extra allocations above the capacity are bound to cause serious concern to the quality of service. Therefore, if the overburdening of the facilities is not penalized, it is bound to cause serious deterioration in the quality of service. The problem considered in this chapter is the same as considered in Chapter 2 except that overburdening is allowed here but a penalty cost. This cost has been linearly related to the amount of overburdening. The penalty cost can be taken as the cost of overtime by doctors, nurses and other staffs for treating extra number of MTBs; or attributed to many similar concerns. By doing so, an attempt has to be made to handle the service issue. The maternal healthcare problem and information related to various facility types, service types and maximum coverage distance discussed in Chapter 2 are followed as it is here also.

3.3 The Mathematical Model

Most of the notations used in the proposed mathematical model in this section are the ones that have also been defined and used in the Section 2.3. The same are repeated here to provide ease in understanding of the proposed model that follows it and represents the planning problem considered in this chapter.

Sets and Indices

I: set of locations of MTBs, $I=\{1,2,...,m\}$, indexed by i

J: set of locations of potential facilities, $J=I$, indexed by j, k and n

L: set of service types offered, $L=\{1,2,3\}$, indexed by l, l' and l''

F: set of types of facility, $F=\{I,II,III\}$, indexed by f and u

Parameters

M: a big number

P: penalty cost per additional MTB allocated beyond the capacity of any service type available at a facility

d_1: a limit on the maximum distance to be covered by an MTB during a non-referral visit

d_2: a limit on the maximum distance to be covered by an MTB during the referral visit

d_{ij}: distance between locations $i \in I$ and $j \in J$

d_{jk}: distance between facility locations $j \in J$ and $k \in J$

C_{ij}: travel cost incurred by an MTB for visiting the facility at a location $j \in J$ from its current location $i \in I$

C_{jk}: travel cost incurred by an MTB for referral visit from current facility location $j \in J$ to a referral facility at location $k \in J$

F_j^f: fixed cost on establishing a facility of type $f \in F$ at location $j \in J$

Q^{lf} : capacity of service type $l \in L$ available with facility type $f \in F$

W_i^l : number of MTBs at a location $i \in I$ requiring service type $l \in L$

$\theta^{ll'}$: proportion of referrals for service type $l' \in L$ from service type $l \in L$, where $l' > l$

$$\alpha_{ij} = \begin{cases} 1, & \text{if a facility for non-referral visit at a location } j \in J \text{ is within the coverage distance of an MTB at a location } i \in I \text{ (i.e., } d_{ij} \le d_1\text{)}, \\ 0, & \text{otherwise} \end{cases}$$

$$\beta_{jk} = \begin{cases} 1, & \text{if a referral facility at a location } k \in J \text{ is within the coverage distance of a lower level facility at location } j \in J \text{ (i.e., } d_{jk} \le d_2\text{)}, \\ 0, & \text{otherwise} \end{cases}$$

Decision variables

x_{ij}^l :number of mothers-to-be at location $i \in I$ being allocated to a facility at a location $j \in J$ to receive service type $l \in L$

x_{jk}^{lm} : number of mothers-to-be receiving service type $l \in L$ at facility $j \in J$ who require referral visit to a facility $k \in J$ for a higher service type $m \in L$

$\bar{x}_j^l$: number of mothers-to-be allocated to a facility at a location $j \in J$ to receive service of type $l \in L$ but beyond the available capacity of that service type

$$y_j^f = \begin{cases} 1, & \text{if a facility of type } f \in F \text{ is located at } j \in J, \\ 0, & \text{otherwise} \end{cases}$$

The mathematical model of the Hierarchical Capacitated Facility Location-Allocation problem with Penalty (HCFLP) is presented hereunder using the above notations.

Minimize $$\sum_{j\in J}\sum_{f\in F} F_j^f y_j^f + \sum_{i\in I}\sum_{j\in J}\sum_{l\in L} c_{ij} x_{ij}^l + \sum_{k\in J}\sum_{j\in J}\sum_{f\in L}\sum_{l'\in L} c_{jk} x_{jk}^{ll'} + \sum_{j\in J}\sum_{l\in L} P\bar{x}_j^l \qquad (3.1)$$

Subject to

$$\sum_{j\in J} x_{ij}^l = W_i^l, \qquad \forall i \in I, l \in L \qquad (3.2)$$

$$\sum_{k\in J} x_{jk}^{ll'} = \theta^{ll'} \left\{ \sum_{i\in I} x_{ij}^l + \sum_{l''\in L}\sum_{n\in J} x_{nj}^{l''l} \right\}, \qquad \forall\, j \in J, l \in L, l', l'' \in L, \text{where } l'' < l < l' \qquad (3.3)$$

$$\sum_{i\in I} x_{ij}^{l} + \sum_{k\in J}\sum_{l'\in L} x_{kj}^{ll'} - \overline{x_j^{l}} \le \sum_{f\in F} Q^{lf} y_j^{f}, \quad \forall\, j\in J, l\in L \tag{3.4}$$

$$\sum_{f\in F} y_j^{f} \le 1, \quad \forall j\in J \tag{3.5}$$

$$x_{ij}^{l} \le \sum_{f\in F} \alpha_{ij} y_j^{f} M^{lf}, \quad \forall i\in I, \forall\, j\in J, \forall l\in L \tag{3.6}$$

$$x_{jk}^{ll'} \le \sum_{f\in F} \beta_{jk} y_k^{f} M^{l'f}, \quad \forall\, j\in J, \forall k\in J, \forall l\in L, \forall l'\in L, l'>l \tag{3.7}$$

$$Q^{2I} = Q^{3I} = Q^{3II} = 0 \tag{3.8}$$

$$x_{ij}^{l}, x_{jk}^{lm}, \overline{x_j^{l}} \ge 0, \quad \forall i\in I,\, j\in J, l\in L, m\in L \tag{3.9}$$

$$y_j^{f} \in \{0,1\}, \quad \forall\, j\in J,\, f\in F \tag{3.10}$$

The flow of MTBs to various types of facilities is the same as provided in Figure 2.1 in Chapter 2. The four expressions in the objective function (3.1) indicate the fixed cost of creating the healthcare facilities, the cost of travelling to the facilities, the referral cost, and the penalty cost, respectively. Constraint (3.2) ensures that the demand is completely met; whereas constraint (3.3) defines the referral of MTB from a lower-level facility to a higher one. Constraint (3.4) is a capacity constraint and it also determines the excess number of MTBs beyond the capacity at a facility for each service type. According to constraints (3.5), only one type of healthcare facility can be established at a given location. Constraint (3.6) indicates that an MTB can only be assigned to a facility type if it is located within the coverage distance, whereas constraint (3.7) states the same for referral cases. The true features of the facility types are captured by constraint (3.8), that shows service types 2 and 3 not being available with facility type I, and

service type 3 not being available with facility type II. The nature of the decision variables is defined by constraints (3.9) and (3.10).

3.4 Metaheuristics Used as Solution Approaches

HCFLP model presented in Section 3.3 is NP-hard. The Indian maternal healthcare facility planning problem may have to address a problem with an excess of 1000 locations. For such a large MILP, finding an optimal solution in a reasonable amount of time may not be possible. For this reason, three metaheuristics are employed in solving the location problem with the binary integer variables only. This information is later used in arriving at allocation decisions using the Gurobi solver. Because of the sequential approach, the size of the allocation problem reduces drastically. It not only helps in reducing the computational time but also results in a better quality solution. The problem associated with the usage of the metaheuristics has been explained, taking the case of one of these metaheuristics in the next section. The metaheuristics used for the proposed planning problem, with values used for their parameters, are presented in the following sub-sections.

3.4.1 Particle swarm optimization

Kennedy and Eberhart (1995) were the first to present Particle Swarm Optimization (PSO), which was inspired by the social behaviour of a flock of birds and their search for food and shelter. PSO is a well-known evolving population-based metaheuristic and is commonly employed for stochastic optimization. A swarm in PSO is made up of a variety of interactive particles that fly in multi-dimensional search space. Each particle represents a potential solution to the problem and has a randomly initialized position and velocity. A particle updates its next position and velocity based on its own experience and that of the swarm. The location of the particle corresponds to a solution, while its quality is by the fitness function (the

objective function). The quality of the solution is improved by following an iterative strategy for updating all the particle's positions and velocities. PSO has been successfully applied for various continuous optimization problems (Ding et al., 2014; Huang et al., 2012; Kayhan et al., 2010; Li et al., 2021). Kennedy and Eberhart (1997) developed a binary version of PSO (binary PSO) for discrete optimization problems. PSO has also been applied by many researchers for combinatorial optimization problems such as lot-sizing problems (Taşgetiren and Liang, 2004), scheduling problems (Chen et al., 2013), vehicle routing problems (Ai and Kachitvichyanukul, 2009; Marinakis et al., 2010; Peng et al., 2019), closed-loop supply chain network (Santander et al., 2020), facility layout planning (Elkady and Abdelsalam, 2016), knapsack problem (Chih et al., 2014) and path optimization (Wang et al., 2021). After going through the literature review by Basu et al. (2015), it is observed that the application of binary PSO (BPSO) to discrete facility location problems is still limited. Sevkli and Guner (2006) and Saha et al. (2011) worked on uncapacitated facility location problems by using a continuous particle swarm optimization algorithm. In the past, many researchers (Ben et al., 2020; Guo et al., 2014; Premalatha and Natarajan, 2008) have worked to improve the performance of PSO using various search strategies such as local search and variable neighbourhood search for discrete problems.

Taking a cue from these research works and the experience gained during experimentation, the parameter values chosen and the steps followed are briefly presented below. The framework of hybridized approach using binary PSO metaheuristic is shown in Figure 3.1.

Step 1: Parameter initialization

The values used for the various parameters are as follows.

Swarm size (i.e., number of particles) = 20, and

Maximum number of iterations = 100.

The parameters in Equation (3.11), presented later, are assigned values detailed below.

Inertia weight parameter (c_1) = 0.1

Cognitive learning parameter (c_2) = 0.1

Social learning parameter (c_3) = 0.1

Step 2: Generation of initial location solution

A greedy heuristic is used to construct the initial solution. First, the overall demand for each service type is determined. The minimal number of CHCs required is then determined depending on the demand for service type 3. Considering the available capacity for service type 2 with these CHCs, the balance demand for service type 2, is determined. Based on this demand and the capacity for service type 2 with PHCs, the number of PHCs to be opened is determined. After this, a similar process is used to work out the number of SCs required considering the amount of availability of service type 1 with opened CHCs and PHCs. The initial solution is worked out by taking 40% more of the required number of facilities of each type to address the coverage distance constraint and to avoid significantly high penalty costs. The facilities thus determined are distributed at random over the possible locations.

Step 3: Feasibility check

Now each particle is taken, and its prospect for leading to a feasible solution is checked from the coverage point of view both for referral and non-referral cases. It may so happen that some locations may not find the required facilities within the coverage distance limits. Out of all these locations, a location is randomly chosen, and the required facility type is established there. Once again, the process of feasibility check is carried out. On encountering infeasibility,

this process is repeated until a feasible solution emerges. This process is repeated for all the particles.

Step 4: Determination of allocation decisions

Corresponding to the locations arrived in the previous step, values for y_j^f variables are determined. Corresponding to y_j^f value as '0', each of the allocation variables x_{ij}^l, $x_{jk}^{ll'}$ and $\bar{x}_j^l$ are also assigned values equal to zero. With this input, the HCFLP model is solved using the Gurobi solver. The value of the objective function, given by expression (3.1), is the value of the fitness function corresponding to a particle. This process is repeated for all the particles.

Step 5: Updatation of particle best and global best solutions

The current fitness value of each particle is compared with the particle's best fitness value registered so far (it is set equal to the current value in the first iteration). If the particle's current position has a higher fitness function value than the prior best, the particle's best position is updated with the current position and the particle's best fitness value, p_{best}, with the current fitness value. Applying this process to all the particles, the current global best value, g_{best}, is determined. After discovering a better current global best value, the previous global best value and the corresponding position are updated.

Step 6: Updating particle velocity and position

The position and velocity of a particle 'p' are defined in $|J|$-dimensional space, where $|J|$ is the number of locations for possible assignment of facilities to them. The value of y_{jp}^{fk} is the value of a variable y_j^f of a particle p in k^{th} iteration. Similarly, v_{jp}^{fk} is the movement to be imparted to particle p in j^{th} direction in k^{th} iteration. The velocity of a particle p as $v_{jp}^{f,k+1}$ in $(k+1)^{\text{th}}$ iteration

is determined using Equation (3.11) r_1 and r_2 are the uniformly distributed random numbers in the range [0,1]. The next position ($y_{jp}^{f,k+1}$) of each particle is determined by adding the velocity ($v_{jp}^{f,k+1}$) to the current position (y_{jp}^{fk}) given by Equation (3.12). In the first iteration, the velocity values for all the particles are kept at zero level.

$$v_{jp}^{f,k+1} = c_1 v_{jp}^{fk} + c_2 r_1 (p_{best,p} - y_{jp}^{fk}) + c_3 r_2 (g_{best} - y_{jp}^{fk}) \tag{3.11}$$

$$y_{jp}^{f,k+1} = y_{jp}^{fk} + v_{jp}^{f,k+1} \tag{3.12}$$

After adding the velocity to the position of the particle using Equation (3.12), the binary variables may take non-binary values causing infeasibility. To deal with this issue, first piece-wise linear function (Equation (3.13)) and then sigmoid function (Equation (3.14)) is used to update the position of the particle. Final position of each particle will be determined using Equation (3.15).

$$y_{jp}^{fk} = \begin{cases} 1, \text{ if } y_{jp}^{fk} \geq 1 \\ y_{jp}^{fk}, \text{if } 0 < y_{jp}^{fk} < 1 \\ 0, \text{ if } y_{jp}^{fk} \leq 0 \end{cases} \tag{3.13}$$

$$sigmoid(y_{jp}^{fk}) = \frac{1}{1 + e^{-y_{jp}^{fk}}} \tag{3.14}$$

$$y_{jp}^{fk} = \begin{cases} 1, \ if \ \ U(0,1) < sigmoid(y_{jp}^{fk}) \\ 0, \ otherwise \end{cases} \tag{3.15}$$

Step 7: Stopping criterion

The process is terminated only if either Steps 3 to 6 have been iterated for a specified number or the total CPU time used has touched or crossed the limit set on it. Otherwise, the control is transferred to Step 3.

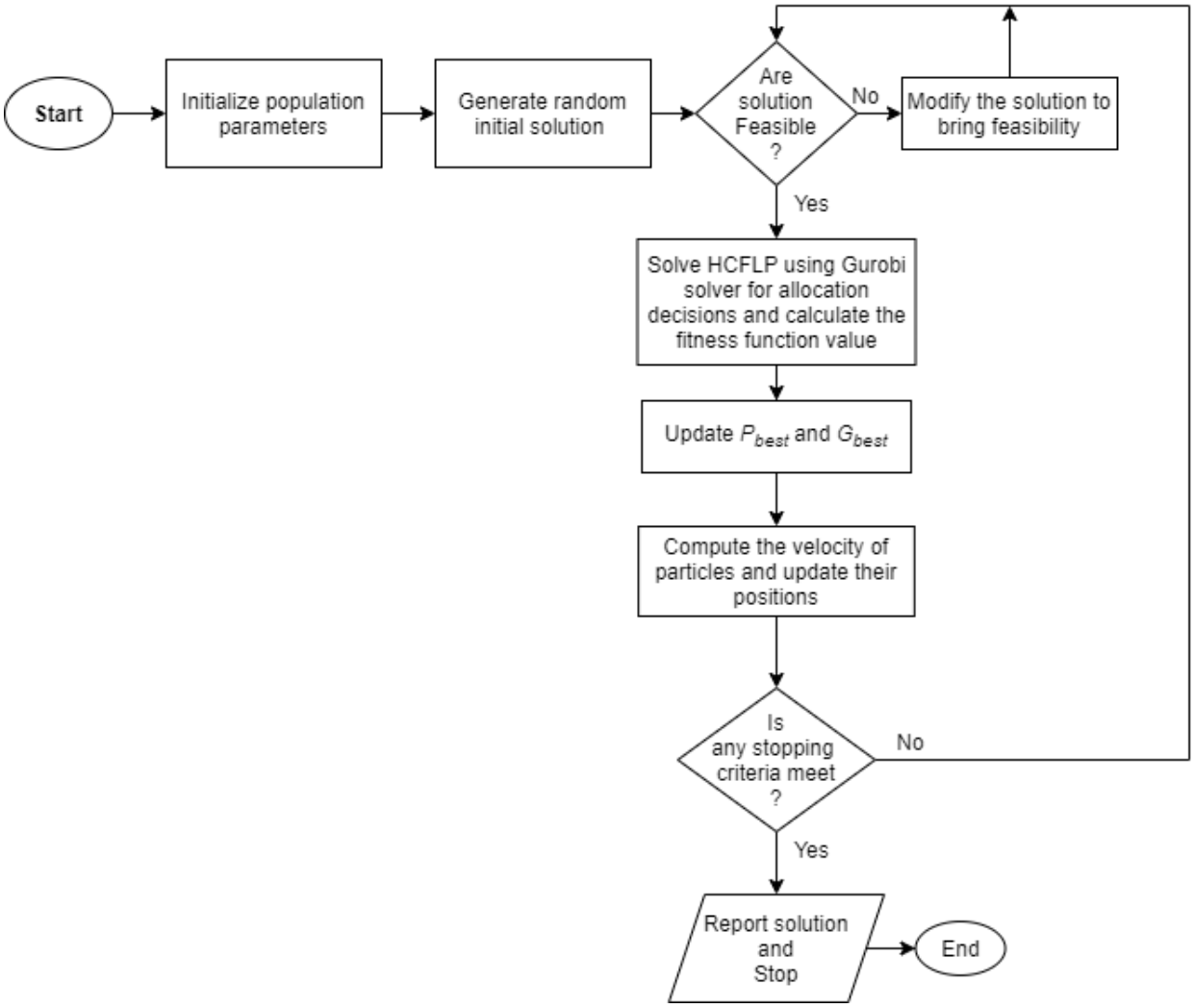

Figure 3.1: Framework of proposed PSO

3.4.2 Artificial bee colony algorithm

Karaboga and Basturk (2007) first introduced the Artificial Bee Colony (ABC) algorithm motivated by the intelligent foraging behaviour of the honey bee swarm. Bees are classified into three categories: (i) employed bees, (ii) onlooker bees and (iii) scout bees. Each employed bee exploits the nectar sources and memorizes the location of her discovered food source. They unload the nectar to the hive and communicate with onlooker bees waiting in the hive. This information about the good quality food source is shared in the form of a dance by the

employed bees. Onlooker bees choose a food source to exploit depending upon the frequency of dance which is proportional to the quality of the food source. If the food source is exhausted, then the source is abandoned, and its employed bee becomes a scout bee to explore a new food source unvisited before. The local search and global search strategies of ABC help in the exploration and exploitation of the search space. Comprehensive detail of the work related to the ABC algorithm is available in the review paper by Akay et al. (2021).

The ABC algorithm has been applied for solving a wide variety of discrete optimization problems, such as facility location (Farahani et al., 2014; Lin et al., 2018; Tuncbilek et al., 2012), knapsack problem (Santana et al., 2019), closed-loop supply chain network problem (Cui et al., 2017), waste collection problem (Wei et al., 2019), scheduling problem (Awadallah et al., 2015; Wang et al., 2015; Xu and Wang, 2021), etc. To solve uncapacitated facility location problems, the ABC metaheuristic was employed by many researchers incorporating efficiency improving strategies, such as the two-phase heuristic (Kashan et al., 2012), XOR-logic (Kiran and Gündüz, 2013), and floor and double mod process (Kiran, 2015) for conversion of continuous values to binary ones. In this work, a piece-wise linear function and sigmoid function (discussed in Section 3.4.1) are used to maintain the binary nature of the variables.

As mentioned at the beginning of this section, the ABC algorithm will be used for solving the location problem and will be hybridized with the Gurobi solver to solve the allocation problems. The ABC algorithm for solving HCFLP is divided into multiple phases initialization, employed bee, onlooker bee, scout bee, and termination phases. The employed and observer bee phases intensify the situation, while the scout bees present a variety of options. Figure 3.2 depicts the ABC algorithm's framework with the other details provided below.

Initialization phase

The ABC algorithm begins with assigning values to its various parameters. Experimenting with several problems, the ABC algorithm was found to perform better for solving HCFLP with parameter values detailed below.

Number of food sources = 20

Maximum number of iterations = 100

Maximum number of exploitations = 10

Initial location solutions are generated following Steps 2 and 3 of the PSO (Section 3.4.1). Based on the location solution, allocation decisions and the corresponding fitness value are worked out following Step 4 of the proposed PSO (Section 3.4.1). A solution is equivalent to a food source, and each food source is managed by a separate employed bee. Naturally, the number of employed bees is equal to the number of food sources.

Employed bee phase

In this phase, each employed bee attempts to identify a source with better quality of food in the vicinity of its current food source. This process is mathematically expressed by Equation (3.16).

$$y_j^{f,k+1} = y_j^{fk} + \phi(y_j^{fk} - y_j^{f,random}) \qquad (3.16)$$

In the above equation, variables y_j^{tk} represent location decisions arrived in k^{th} iteration, variables $y_j^{f,random}$ represent randomly identified neighbourhood locations with food, and variables $y_j^{f,k+1}$ to represent new food sources (locations for various facilities) in the

neighbourhood of y_j^{fk} at $(k+1)^{th}$ iteration. ϕ is the uniformly distributed random number in the range of [-1, 1]. To ensure the binary nature of location variables y_j^f , Equations (3.13), (3.14) and (3.15) are used. After this, Step 3 of the proposed PSO approach is applied to ensure the feasibility of the location solution. After having feasible location solutions, Step 4 of the PSO algorithm is used to find allocation decisions and the corresponding fitness value. If the new solution is better than the old one, the counter is reset to zero, and the new solution is memorized. Otherwise, the food source's counter is incremented by one. This procedure is carried out for each employed bee.

Onlooker bee phase

Initially, onlooker bees wait in the hive and choose to fly to a good quality food source, according to the information shared by the employed bees. The probability of selecting k^{th} food source is calculated as follows:

$$P_k = \frac{Obj_k}{\sum_{n=1}^{N} Obj_n} \tag{3.17}$$

where P_k is the chance of the k^{th} solution getting selected by an onlooker bee and Obj is the fitness value of k^{th} solution. These probability values use the roulette wheel selection technique that provides a better chance of selection of good quality food sources (solutions) and low probability for low-quality solutions. After an employed bee chooses a food source, the onlooker bee also looks for a better food quality source by adopting Equation (3.16). This process is the same as what was adopted by the employed bees. If the solution identified by the onlooker bee is better than the employed bee's solution, the employed bee memorizes the

new solution. Otherwise, the food source's counter is incremented by one. This procedure is also carried out for each onlooker bee.

Scout bee phase

During this phase, each food source's counter is put to the test. The counters with the highest content for food sources are found. The employed bee of this food source becomes a scout bee if the counter is higher than the limit. For this scout bee, a new feasible solution is generated at random (using Steps 2 and 3 of the proposed PSO). The food source's counter is reset, and the scout bee is reclassified as an employed bee.

Termination criterion

The phases discussed above are iterated until the specified number of iterations have been performed or the CPU time used is greater than or equal to the limit put on it. The best solution obtained so far is finally reported.

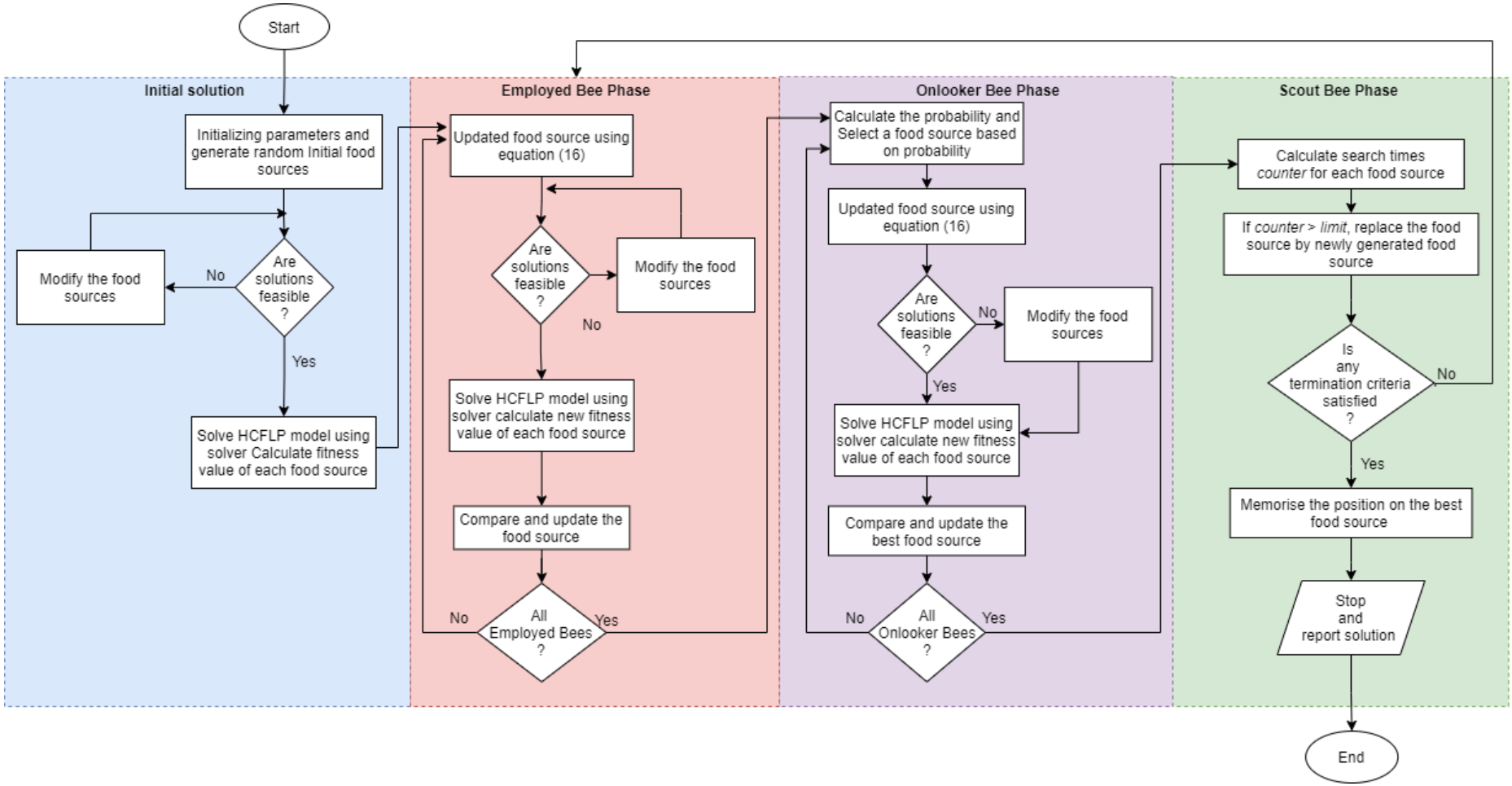

Figure 3.2: Framework for implementation of ABC algorithm

3.4.3 JAYA algorithm

The majority of metaheuristics require numerous controlling parameters, such as mutation probability, crossover probability, and selection operation in the Genetic algorithm, cognitive and social parameters in the PSO, and the number of scout bees, onlooker bees, and employed bees in the ABC algorithm. Tuning these parameters is a time-consuming and tedious task, and sometimes the range of the parameters may vary depending on the complexity and nature of the problem. Rao et al. (2016) introduced the Jaya algorithm, which requires no algorithm-specific parameters and is simple in implementation. JAYA is a population-based algorithm and is similar to other metaheuristics. It aims to move towards the best solution and away from the worst one after each iteration. In recent years, JAYA has witnessed a growth in its popularity due to its ability to identify quality solutions in lesser time. Abundant applications of it can be found in the literature for a variety of problems, such as surface grinding process parameter identification (Rao et al., 2016), facial emotion recognition (Wang et al., 2018), fracture mechanics (Khatir and Wahab, 2019), crack identification (Khatir et al., 2020), job shop scheduling (Caldeira and Gnanavelbabu, 2019; Gunduz and Aslan, 2021; He et al., 2021). Prakash et al. (2017) developed the first binary version of the JAYA algorithm to solve the problem of phasor measurement units related to the power system. Chaudhuri and Sahu (2021) employed a binary JAYA algorithm for the feature selection approach for data classification. Aslan et al. (2019) compared the binary JAYA algorithm with particle swarm optimization, artificial bee colony algorithm, binary tree-seed algorithm and genetic algorithm for uncapacitated facility location-allocation model. They found that the binary JAYA algorithm performs much better compared to other ones in terms of solution quality and robustness.

Houssein et al. (2021) and Zitar et al. (2022) have come out with a review on the JAYA algorithm.

Similar to binary PSO and ABC, the binary JAYA algorithm has also been used to determine the best possible locations for facilities. The framework of the proposed binary JAYA algorithm is shown in Figure 3.3, and various steps are as follows:

Step 1: Initialize the algorithm with a population size of 10 and a maximum number of iterations equal to 100.

Step 2: Generate random initial solutions whose number would be equal to the population size. The initial solutions (locations of facilities) are obtained using the greedy approach detailed in Step 2 of the proposed PSO.

Step 3: A feasibility check is carried out for the solution of each candidate. If the solution found is not feasible, then the candidate solution is repaired using Step 3 of the proposed PSO. These feasible locations of facilities determine the value of y_{jc}^{f}.

Step 4: Based on the values of variables y_{jc}^{f}, allocation decisions are made by solving the HCFLP model using the Gurobi solver. The fitness value for each candidate is determined using expression (3.1).

Step 5: If the obtained fitness value is found to improve (initially set to very high), then update the corresponding candidate solution; Otherwise, the solution is retained as such.

Step 6: Identify the J_{best} and J_{worst} in the population and update the candidates' solution using Equation (3.18).

$$y_{jc}^{f,k+1} = y_{jc}^{fk} + r_1(y_{jbest}^{fk} - |y_{jc}^{fk}|) - r_2(y_{jworst}^{fk} - |y_{jc}^{fk}|) \tag{3.18}$$

Where r_1 and r_2 are uniformly distributed random numbers generated in the range of [0,1], and y_{jbest}^{fk} and y_{jworst}^{fk} are the best and worst values of the variables from best fitness value and worst fitness value. The updated solution may take non-binary values, leading to an infeasible solution. Equations (3.13), (3.14) and (3.15) are to be used to deal with this issue, as in the case of the proposed PSO.

Step 7: Steps 3 to 6 are repeated until the specified number of iterations have been performed or the total CPU time used is greater than or equal to the limit put on it.

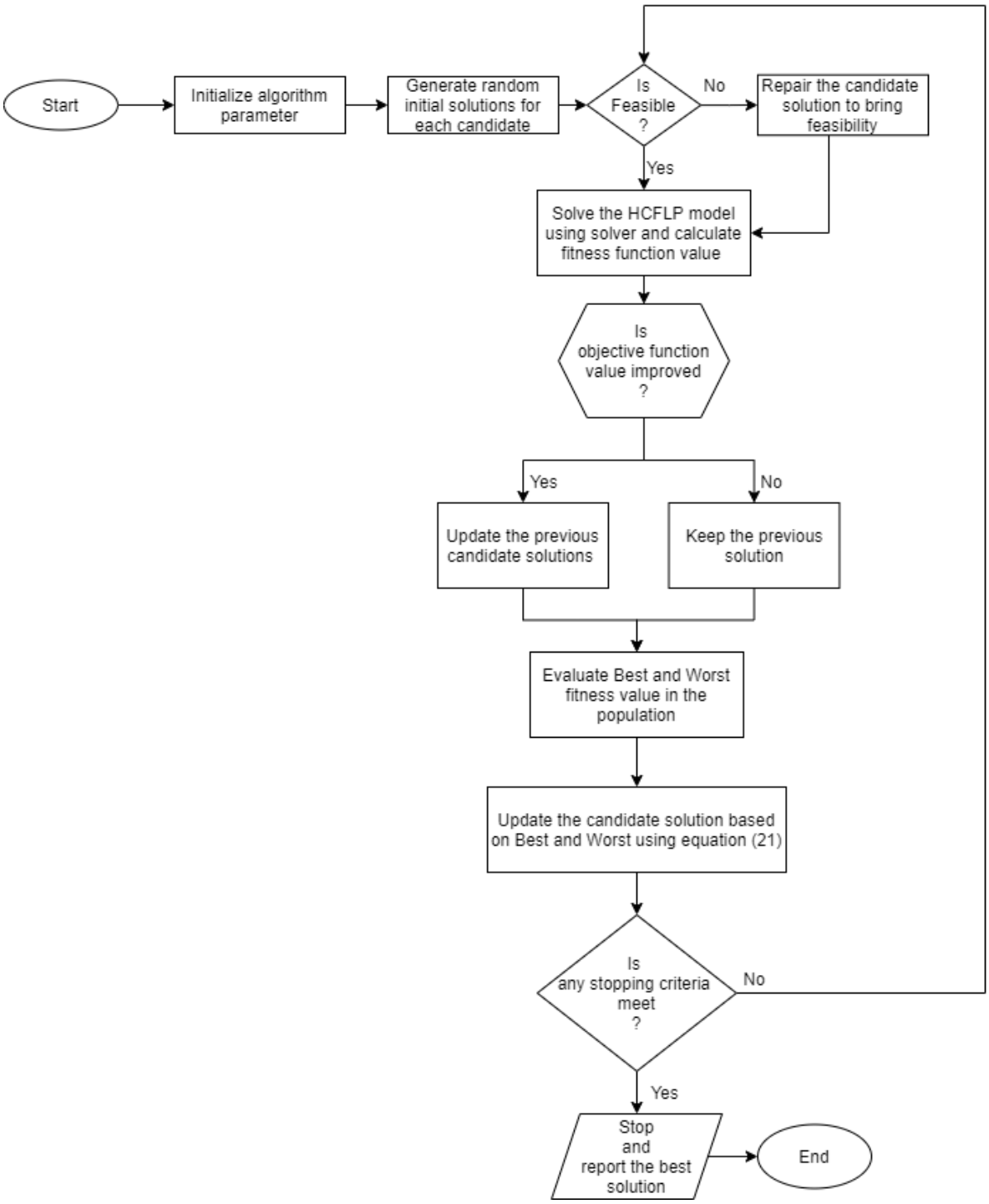

Figure 3.3: Framework for the proposed JAYA algorithm

3.5 Performance Study of the Proposed Metaheuristics

The findings of the proposed metaheuristics' and comparative performance are reported in this section. For solving the HCFLP mathematical model, Gurobi optimization solver with Python 3.8 is used. Besides, the metaheuristics were also coded in Python 3.8. Extensive computational experiments were carried out on the supercomputer "PARAM-SHIVAY" with a configuration of 2* Intel Xeon SKL G-6148, 4 core, 2.4GHz, 192 GB DDR4 2666 MHz memory. In the following subsections, the scheme for generating test instances and, thereafter, computational results are presented.

3.6.1 Generation of test instances

For a fair performance comparison of the considered metaheuristics, the maternal healthcare network planning problems were taken with varying numbers of locations ranging from 100 to 1000 with step size of 100. Five instances of each of these 10 different sizes of the problem (a total of 50 problems) were generated. Based on the total number of binary and continuous variables in HCFLP formulation, the problems were grouped into three categories (i.e., small, medium, and large), as shown in Figure 3.4. From this figure and also from the HCFLP formulation, it can be seen that the number of binary variables increases with an increase in problem size; binary variables increase linearly, and continuous variables exponentially. The demand nodes (or MTB locations), referral proportion, various types of costs and other parameters are similar as discussed in Section 2.5.1 of Chapter 2.

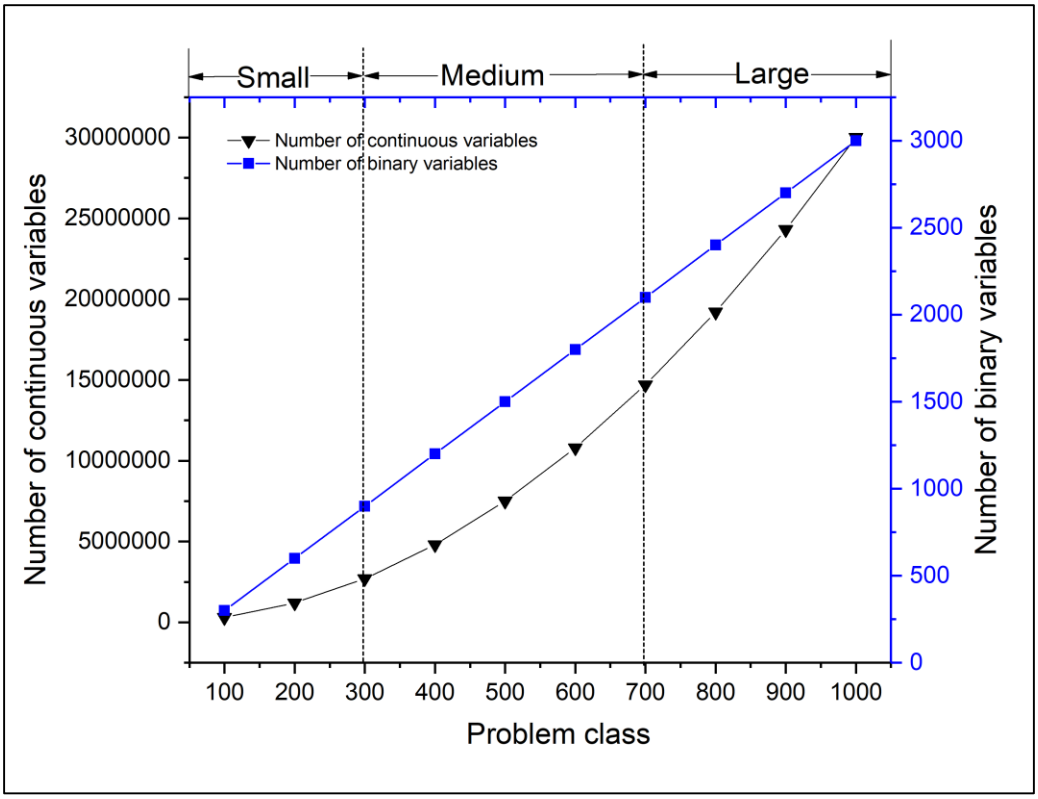

Figure 3.4: Various problem classes

3.6.2 Experimental results and discussion

Table 3.1 shows the experimental results from the use of the three metaheuristics discussed in Section 4. The 'Gap' column reflects the difference between the objective function's upper bound and lower bound values. Columns labelled as 'D_{PSO}', 'D_{ABC}' and 'D_{JAYA}' respectively represent the percentage deviations by which the UB values are less than the corresponding objective function values obtained from PSO, ABC and JAYA algorithms. The comparison was made based on the UB as it corresponds to a feasible solution for the cost minimization problem. The negative values of 'D_{PSO}', 'D_{ABC}' and 'D_{JAYA}' represent the cases where the proposed metaheuristics perform better in comparison to the direct use of HCLFP formulation for problem-solving.

Table 3.1: Experimental results of proposed solution approaches

Problem Size	Problem class	Problem Number	Instance number	Gurobi				PSO			ABC			JAYA		
				LB	UB	Gap (%)	Time (in sec)	Objective value	D_{PSO} (%)	Time (in sec)	Objective value	D_{ABC} (%)	Time (in sec)	Objective value	D_{JAYA} (%)	Time (in sec)
Small	**100**	1	1	10843322	11043534	1.81	86400	11186966	1.30	3963	12135246	9.89	285	12021657	8.86	4117
		2	2	10775075	10908626	1.22	86400	11090778	1.67	17665	11979378	9.82	1908	11842759	8.56	4927
		3	3	11121055	11224319	0.92	86400	11762999	4.80	61022	12096729	7.77	1898	12221136	8.88	4791
		4	4	10724508	10858138	1.23	86400	11010658	1.40	15191	11942642	9.99	2590	11902609	9.62	2860
		5	5	12297042	12521579	1.79	86400	12780545	2.07	6740	13684676	9.29	1881	13856738	10.66	4866
Small	**200**	6	1	20921405	21220988	1.41	86400	22837798	7.62	84173	22958203	8.19	8444	22988992	8.33	18729
		7	2	21498404	21779771	1.29	86400	23400781	7.44	56756	23923684	9.84	8301	23422340	7.54	17975
		8	3	21516587	21856960	1.56	86400	23340975	6.79	35324	23614708	8.04	8334	24046036	10.02	13573
		9	4	20625501	20935964	1.48	86400	22453940	7.25	83609	22653419	8.20	8378	22571620	7.81	31644
		10	5	21062565	21270401	0.98	86400	22818327	7.28	59560	22974466	8.01	8366	23503271	10.50	34442
Small	**300**	11	1	32141797	32707430	1.73	86400	34736001	6.20	61543	35106868	7.34	5024	36478766	11.53	29964
		12	2	31058579	31499761	1.40	86400	33524994	6.43	83235	33903206	7.63	12980	36581694	16.13	32221
		13	3	31348148	35630209	12.02	86400	33992991	-4.60	80032	34398151	-3.46	12963	35774523	0.41	33737
		14	4	31663513	40866909	22.52	86400	34406498	-15.81	34123	35006280	-14.34	13111	37161855	-9.07	25777
		15	5	32343927	32807675	1.41	86400	35084009	6.94	63175	35588164	8.48	12969	36645190	11.70	26018
Medium	**400**	16	1	42182034	54027924	21.93	86400	45865206	-15.11	15835	47001734	-13.00	7930	47923295	-11.30	27560
		17	2	42455783	66495495	36.15	86400	46659708	-29.83	1850	46386089	-30.24	8241	48968111	-26.36	31296
		18	3	42587304	72037318	40.88	86400	46285813	-35.75	30250	46762302	-35.09	8209	47370356	-34.24	22381
		19	4	41534464	67201336	38.19	86400	45303842	-32.58	58180	45796347	-31.85	8142	48634319	-27.63	25161
		20	5	42443462	75307331	43.64	86400	46331898	-38.48	59064	47380585	-37.08	8151	48198355	-36.00	28908
Medium	**500**	21	1	99429394	117434395	15.33	86400	107514373	-8.45	3881	107405934	-8.54	17320	123263678	4.96	50633
		22	2	94868752	123272403	23.04	86400	102954993	-16.48	3967	103068198	-16.39	18392	117643614	-4.57	39071
		23	3	95028497	125492523	24.28	86400	102872347	-18.03	3934	103050991	-17.88	18111	113507038	-9.55	49851
		24	4	101447173	130734355	22.40	86400	109493362	-16.25	3964	109879449	-15.95	17275	122540493	-6.27	58595
		25	5	95642758	125733166	23.93	86400	103459154	-17.72	4128	103535577	-17.65	17184	116862502	-7.06	26221
Medium	**600**	26	1	117288370	145041056	19.13	86400	126841069	-12.55	7577	127017005	-12.43	34275	138618214	-4.43	3549

		27	2	120200797	157385126	23.63	86400	130130341	-17.32	7213	130079885	-17.35	33491	143725719	-8.68	53155
		28	3	116465441	152977610	23.87	86400	126165371	-17.53	7153	126195925	-17.51	32965	141607351	-7.43	3847
		29	4	114179919	144291303	20.87	86400	123722398	-14.26	9332	124145531	-13.96	34907	138718357	-3.86	3750
		30	5	116645215	149457896	21.95	86400	126290271	-15.50	7346	126330094	-15.47	35297	141159922	-5.55	3770
		31	1	131926186	172612885	23.57	86400	142618599	-17.38	11526	142912890	-17.21	52600	157143483	-8.96	5168
		32	2	140434881	172965066	18.81	86400	152068801	-12.08	11758	152213200	-12.00	50520	158762563	-8.21	5158
Medium	**700**	33	3	-	-	-	86400	145314567	-	12009	145471781	-	53454	159441088	-	5323
		34	4	-	-	-	86400	141888496	-	11756	142360817	-	53639	157155157	-	5485
		35	5	132566398	171012083	22.48	86400	143582206	-16.04	12581	143513930	-16.08	54215	161242107	-5.71	5686
		36	1	-	-	-	86400	173014327	-	5728	173094464	-	86400	176045179	-	8819
		37	2	-	-	-	86400	167406007	-	22362	167414155	-	86400	182169480	-	9629
Large	**800**	38	3	-	-	-	86400	162770805	-	5738	162584467	-	86400	179742436	-	9062
		39	4	-	-	-	86400	168226932	-	22190	169361398	-	86400	181863643	-	10324
		40	5	-	-	-	86400	164256117	-	20942	164305481	-	86400	177631335	-	9373
		41	1	-	-	-	86400	189418697	-	34211	189959482	-	86400	203014746	-	13425
		42	2	-	-	-	86400	188886887	-	36449	190841199	-	86400	204066943	-	13605
Large	**900**	43	3	-	-	-	86400	192310717	-	33929	192913614	-	86400	207678891	-	13351
		44	4	-	-	-	86400	190715421	-	34480	191274870	-	86400	205106538	-	14492
		45	5	-	-	-	86400	188464376	-	34635	189204413	-	86400	202956487	-	14958
		46	1	-	-	-	86400	207517807	-	13191	207803503	-	86400	221347849	-	18591
		47	2	-	-	-	86400	211216259	-	13488	211984385	-	86400	226436741	-	18111
Large	**1000**	48	3	-	-	-	86400	208749072	-	14839	209573809	-	86400	222120204	-	19450
		49	4	-	-	-	86400	205800806	-	14219	205991666	-	86400	221505756	-	20430
		50	5	-	-	-	86400	215114994	-	13405	214516128	-	86400	227484427	-	18578

For each problem instance and each solution approach, a maximum CPU time limit of 24 hours was imposed for experimentation purposes. In the straight use of the HCFLP formulation (using the Gurobi solver), no feasible solution was found for the 17 problem instances (33, 34 and 36 to 50) within the specified CPU time limit. For the remaining 33 problem instances (problem number 1 to 32, and 35), feasible but not optimal solutions were obtained., The inability of the Gurobi solver in yielding even feasible solutions for most of the medium-size and all large-size problem instances indicates the ineffectiveness of the solver in handling real large-size problems in the Indian context.

In contrast to the direct use of HCFLP formulation, the proposed metaheuristic approaches (PSO, ABC and JAYA) performed much better and provided good quality feasible solutions, and that too much earlier than the permitted maximum limit of 24 hours on CPU time. The comparison of various solution approaches is presented in subsequent sub-sections.

General observations

The direct use of the HCFLP formulation was found to be advantageous for solving the small size problem instances. For location problems up to 200 nodes, the solver provided a much better solution in much less time. The solutions from metaheuristics were costlier but by less than or equal to 10.66%. For 300 location problems, mixed results were obtained in terms of solution quality. However, the CPU time requirement was lesser than that for the solver. For all the medium-sized problems, the proposed approaches yielded a much better solution than the solver and outperformed the solver in terms of used CPU time. As indicated earlier, for the large-size problem, the solver failed to yield a solution in a specified time limit of 24 hours. In contrast, the proposed metaheuristics could do so in much lesser time.

3.6.3 Comparative performance analysis of proposed metaheuristics

Table 3.1 did not help to provide a perfect and clear superiority of one approach over the other, either based on solution quality or CPU time requirement. For this reason, Wilcoxon signed-rank statistical tests were performed for both CPU time and solution quality for all four approaches. Each of the approaches (approach 1) is compared with the other three approaches (approach 2). The null hypothesis claims that approach 1 has a performance equal to that of approach 2. Hence the alternate hypothesis states that the performances of the two approaches are at variance and thus significantly different. Rejection of the null hypothesis will mean non-rejection of the alternate hypothesis. Non-rejection of the alternate hypothesis will lead to the conclusion of going in favour of that approach, in whose case better results were obtained a greater number of times. The tests were conducted for a significance level (p) of 0.05.

The results shown in Table 3.1 indicate that the quality of solution yielded by the ABC algorithm is quite close to that of the PSO algorithm for most of the problem instances. For the large size problem instances, the PSO approach outperforms the ABC approach in terms of solution quality. The quality of solutions yielded by the JAYA algorithm is the worst in many of the cases. For a confirmatory finding, Wilcoxon signed-rank tests were conducted. The results of the same are presented in Table 3.2.

Table 3.2 clearly shows the metaheuristics to outperform the Gurobi solver in terms of the solution quality. Among the metaheuristics, the domination of the PSO approach over the ABC and JAYA approaches can also be noticed in terms of solution quality. In a comparison of the ABC approach with the JAYA approach, the ABC approach is found to be better than the JAYA approach. The results of the computational experiments and statistical analyses indicate

that, in terms of solution quality, the PSO algorithm generally outperforms the ABC and JAYA algorithms, and the ABC algorithm is better than the JAYA algorithm.

Table 3.2: Wilcoxon-signed rank test for solution quality

Approach 1	Approach 2	Number of instances in which the approach below is better		*p*-value	Remark
		Approach 1	Approach 2		
Gurobi Solver	PSO	13	20	0.00086	Significant*
Gurobi Solver	ABC	13	20	0.0012	Significant*
Gurobi Solver	JAYA	15	18	0.0057	Significant*
PSO	ABC	44	6	0.00001	Significant
PSO	JAYA	50	0	0.00001	Significant
ABC	JAYA	45	5	0.00001	Significant

***** Only those problem instances are taken in the analysis for which Gurobi Solver resulted in a feasible solution within the specified limit on the CPU time.

Table 3.3 summarizes the findings of the statistical analyses on CPU time. This table indicates that the CPU time requirements of the proposed metaheuristics are significantly statistically different ($p < 0.05$) and is lesser than that for the Gurobi solver in all the cases. Because of the statistically significant difference between approaches 1 and 2, the null hypotheses are to be rejected. Further, due to the lesser CPU time requirement, it can be concluded that the proposed metaheuristics surpass the Gurobi solver in terms of CPU time.

Table 3.3: Wilcoxon-signed rank test for CPU time

Approach 1	Approach 2	Number of instances in which the approach below is better		*p*-value	Remark
		Approach 1	Approach 2		
Gurobi Solver	PSO	0	50	0.00001	Significant
Gurobi Solver	ABC	0	50	0.00001	Significant
Gurobi Solver	JAYA	0	50	0.00001	Significant
PSO	ABC	31	19	0.07346	not significant
PSO	JAYA	34	16	0.00880	Significant
ABC	JAYA	24	26	0.00804	Significant

The CPU time requirement of metaheuristics was also compared to one another. There is no significant difference in CPU time requirements of PSO and ABC approaches at $p = 0.05$. However, the resulted p-value of 0.07346 is not very high either. Even for a low p-value of 0.074, The PSO algorithm can be taken to perform better than the ABC algorithm. CPU time requirement by the PSO approach can be observed to be generally lower than that for the JAYA algorithm, and the difference to be statistically significant. Thus, the PSO approach can be taken to be better than the JAYA approach. Similarly, the JAYA approach can be observed to perform better than the ABC approach in the case of a greater number of problem instances. The difference in the CPU time requirements for JAYA and ABC both are found to be statistically significant. In terms of CPU time, this demonstrates the superiority of the JAYA algorithm over the ABC algorithm.

From the foregoing discussions, it is clear that the PSO outperforms all the other three approaches in terms of both solution quality and CPU time.

3.5 Impact of change in demand and penalty

It is expected that an increase or decrease in total demand may require an increase or decrease in the number of facilities to be established. However, due to the consideration of penalty cost on overburdening, the model may seek the balance between the cost of establishing the new facility and the total penalty cost. This section analyses the effect of variation in demand and penalty cost on the number of facilities to be established and on the total cost. For this purpose, demand is varied from -50% to +50% of the demand in the step of 10% resulting in 11 demand levels. For each demand level, 4 different penalty costs taken are 100, 1000, 10000 and 100000, resulting in a total of 44 scenarios. For the purpose of analysis, problem number 14 (D50_4)

from Chapter 2 is taken. The impact of change in the demand and penalty is summarised in Table 3.4.

From Table 3.4, it is observed that when the penalty cost is low, the number of facilities to be established is less, and facilities are overburdened. With the penalty remaining constant, the increase in the demand may not ask for the opening of additional facilities unless the facility establishment cost becomes more economical than the resulting penalty cost. From the above, it can be concluded that the proposed model will always try to seek an optimal balance between the cost of overburdening and establishing new facilities in determining the optimal solution.

The above experimentation also shows that the demand has a lot of impact both on facility establishment cost and penalty cost. The proposed model assumes the demand to be deterministic. But it varies around its mean value in real life, sometimes being too small and some other time being too high, and all randomly. Under such circumstances, the optimal solution from the model may not be truly optimal. For proper decision-making under stochastic environment, the Monte Carlo simulation will be the most appropriate tool. This tool could be used to determine the optimal network of healthcare facilities since maternal healthcare is a sensitive issue and cannot be viewed only an economic perspective. Here the quality of service being provided to MTBs becomes very important for maternal healthcare planning. The simulation approach and the related findings are presented in the next sub-section.

Table 3.4: Impact of change in demand and penalty

Variation in demand	Penalty	Number of established			Total overburdening on			Objective value
		SC	PHC	CHC	Service type 1	Service type 2	Service type 3	
0.5	100	2	0	3	248	0	0	3417874
0.5	1000	3	0	3	0	0	0	3476116
0.5	10000	3	0	3	0	0	0	3476116
0.5	100000	3	0	3	0	0	0	3476116

0.6	100	3	0	3	598	462	0	3619954
0.6	1000	3	1	3	0	0	0	3982202
0.6	10000	3	1	3	0	0	0	3982202
0.6	100000	3	1	3	0	0	0	3982202
0.7	100	4	0	3	948	1139	0	3838268
0.7	1000	4	1	3	0	139	0	4245413
0.7	10000	3	2	3	0	0	0	4487591
0.7	100000	3	2	3	0	0	0	4487591
0.8	100	6	0	3	298	1817	0	4053797
0.8	1000	4	2	3	0	0	0	4611816
0.8	10000	4	2	3	0	0	0	4611816
0.8	100000	4	2	3	0	0	0	4611816
0.9	100	7	0	3	647	2494	31	4274412
0.9	1000	4	3	3	47	0	31	5187210
0.9	10000	5	3	3	0	0	31	5501937
0.9	100000	4	2	4	0	0	0	5592217
1	100	8	0	3	997	3171	134	4503366
1	1000	6	3	3	0	171	134	5620288
1	10000	6	2	4	0	0	0	5799918
1	100000	6	2	4	0	0	0	5797499
1.1	100	9	0	3	1347	3848	238	4732114
1.1	1000	6	4	3	0	0	238	6054858
1.1	10000	6	3	4	0	0	0	6310906
1.1	100000	6	3	4	0	0	0	6305515
1.2	100	11	0	3	696	4525	341	4960937
1.2	1000	6	5	3	0	0	341	6654118
1.2	10000	6	4	4	0	0	41	7211371
1.2	100000	6	3	5	0	0	0	7289378
1.3	100	12	0	3	1046	5202	445	5190368
1.3	1000	7	4	4	0	2	145	7078844
1.3	10000	7	3	5	0	0	0	7413272
1.3	100000	7	3	5	0	0	0	7415501
1.4	100	13	0	3	1396	5879	548	5422352
1.4	1000	8	6	3	0	0	548	7572922
1.4	10000	7	4	5	0	0	0	7901832
1.4	100000	7	4	5	0	0	0	7901991
1.5	100	14	0	3	1746	6556	652	5654806
1.5	1000	8	7	3	0	0	652	8171941
1.5	10000	8	3	6	0	0	0	8520099
1.5	100000	8	3	6	0	0	0	8515690

3.6 Hybrid Optimization-Simulation Framework with Normally Varying Demand

The formulation proposed in Section 3.3 assumes the demand to be stationery. In the real world, the daily demand cannot be so. Under this situation, the HCFLP model will have use mean demand values. However, working with the mean demand will not give the right picture of the actual service quality. The demand in excess of the capacity on a day can be much more than what will be resulted from the application of the proposed model. The quality is expected to be poorer with the increasing level of variability in demand. To have a fair idea about the service quality, a hybrid optimization-simulation framework is proposed. The generation of the initial solution and further optimization process are discussed in the optimization phase, and the Monte-Carlo method used for the simulation purpose is discussed in the simulation phase.

3.6.1 Optimization phase

The HCFLP model can be solved optimally with a state of the art solver like Gurobi for small size problem instances. However, as the problem size increases, the time required to solve the problem increases exponentially due to its combinatorial nature resulting from the binary variable y_j^f , which denotes the opening of a facility type $f \in F$ at a location $j \in J$. Therefore, obtaining even a good quality solution for medium and large size problems is a challenge. Therefore, we propose an equivalent set covering model to obtain an initial solution by relaxing the travelling cost, referral cost, and penalty from the objective function. Further, a set covering constraint (3.20) is added to the model, which ensures that at least one facility should be opened within the coverage distance. The constraint (3.20) is redundant, but it

provides a good quality initial solution for the considered problem. The set covering model for the considered problem is as given below.

$$\textbf{Minimize} \sum_{j \in J} \sum_{f \in F} F_j^f y_j^f \tag{3.19}$$

$$\sum_{j \in J} \alpha_{ij} y_j^f \geq 1, \quad \forall i \in I, f \in F \tag{3.20}$$

and (3.2), (3.3), …, (3.10).

The initial solution provided by the set covering model is fed as a starting solution to the black box solver Gurobi. The black box solver gives the optimal solution for allocation decisions within a reasonable time limit. Further, this solution is used for the Monte-Carlo simulation.

3.6.2 Simulation phase

A Monte-Carlo simulation framework is discussed here in conjunction with the proposed HCFLP model. The solution of the HCFLP model from the optimization phase (location of opened facilities and allocation details, including referrals) will be fed into the simulation model to study the effect of randomness in demand. Thus, the locations of the opened facilities, allocation of MTBs to the facilities, and referrals proportions will be kept fixed while allowing the number of MTBs at each location to follow the Normal distribution with the mean as the demand value used in the proposed formulation for the given problem. The steps involved in the Monte-Carlo simulation are as follows:

Step (1): For a specified coefficient of variation (COV)value, the standard deviation value is determined as COV times the mean value. With the mean and standard deviation known, the Normally distributed demand from various locations for various service

types can be generated. Assuming that the variability is not going to be location specific and its extent would be the same, COV for the demand is to be the same for each location and service type.

Step (2): After generating the number of MTBs at a location, its allocations to various facilities will be carried out separately for each service type in proportion to what was yielded by HCFLP. For the example of Chapter 2, the HCFLP model distributed the demand of 368 MTBs for service type 1 from location 9 in the proportion of 10:358 between SCs at locations 2 and10, respectively. If the randomly generated demand is 1.5 times of 368, then 15 (1.5 times of 10) MTBs will be sent to SC at location 2 and 537 (1.5 times of 358) to SC at location 10. A similar approach will be followed in distributing the referrals.

Step (3): Once the allocations have been worked out, the allocations beyond the capacity can be worked out. Having determined the excess number of MTBs for each level at each facility, the service level for all the three types of services is determined using the following relationship.

$$S_l = \frac{D_l - E_l}{D_l} \tag{3.21}$$

where,

S_l : Service quality of a service type l

D_l : Total of the demand for the service type, including referrals, and

E_l : Total of the excess allocations made to units of service type l.

Step (4): Step 2 and 3 are repeated for initial independent replications (R_0) and the total number of independent replication (R) required are determined using Equation (3.22) (Banks *et al.*, 1998).

$$R \geq \left(\frac{z_{\alpha/2} S_0}{\varepsilon} \right)^2 \tag{3.22}$$

where $Z_{\alpha/2}$ is the $100(1-\alpha/2)$ percentage point of the standard normal distribution. $\in$ is the pre-specified error criterion and S_0 is the standard deviation of initial replication (R_0).

Step (5): Steps 2 and 3 are followed for an additional number of independent replication (R - R_0) and calculate the mean service level for a 95% confidence interval. The half-length (h.l.) for the 95% confidence interval is computed using Equation (3.23).

$$\text{h.l.} = t_{\alpha/2,R-1}\sigma(\theta) \tag{3.23}$$

where $\sigma(\theta) = S/\sqrt{R}$, S is the standard deviation, and R is the total number of replications.

The optimization and simulation framework is shown in Figure 3.5. The right side block shows the optimization framework, and the left side block shows the simulation framework. For the optimization, various problem instances with a known number of MTB (W_i^l) at a location are solved. First, the initial solution is determined using a set covering model and then that solution is further optimized using Gurobi solver. This solution is now used for simulation. In the simulation, normally distributed number of MTBs at each location are determined using the coefficient of variation (COV) and then the same are allocated according to the allocation

solution determined in optimization phase. This process is repeated or replicated until a pre specified amount of precision on service quality is achieved.

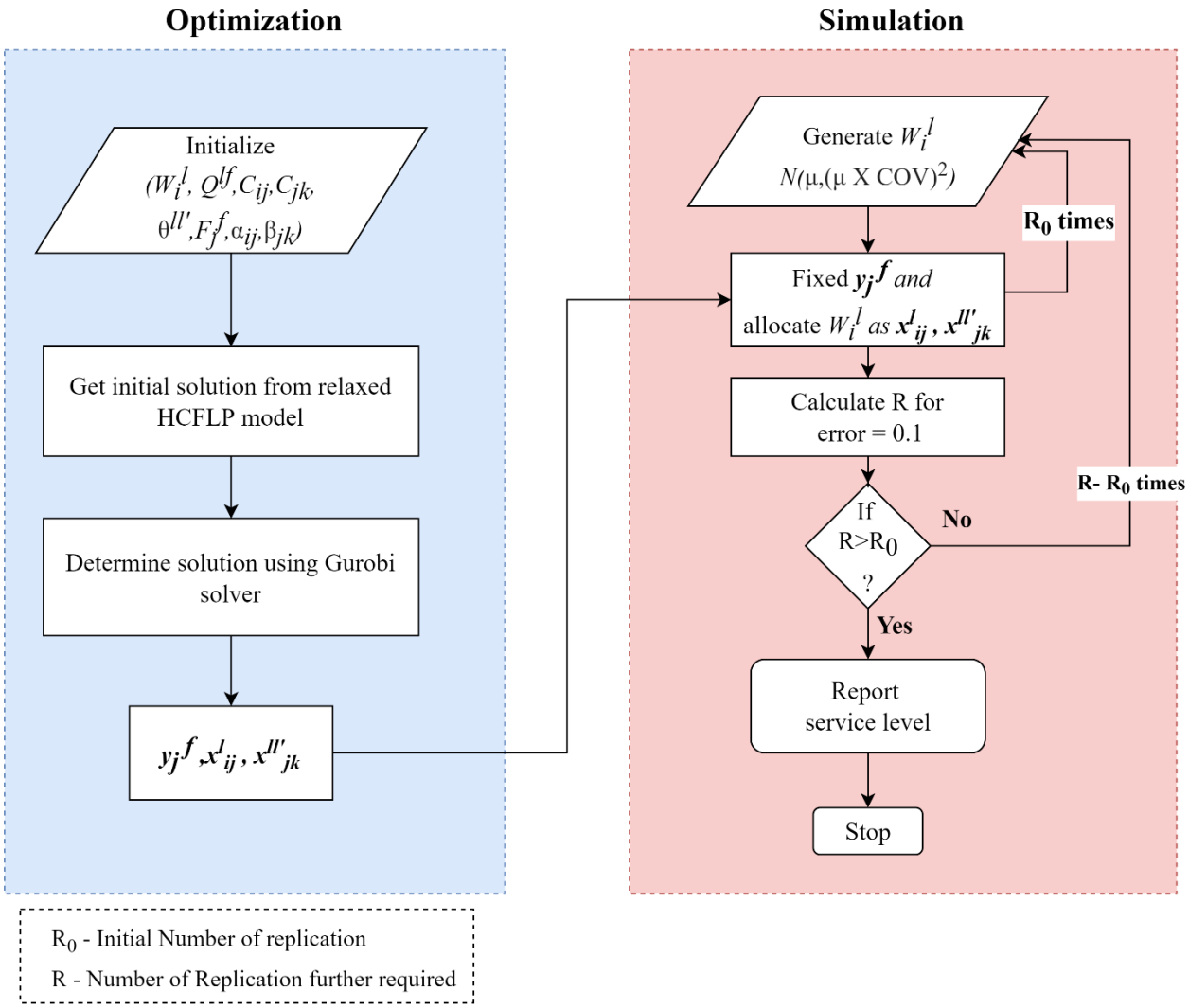

Figure 3.5: Optimization and simulation framework

The simulation model was validated using face validity and validation based on Input-Output Transformations (Banks *et al.*, 1998). For face validity, experts were involved. Validation of input-output transformations was carried out by visualizing the impact on the service quality, which was found to decrease with the increase in the demand level or its variability or with the

reduction in the capacity of various facilities for various service types. The results of such transformation, for the example problem of Chapter 2 (Section 2.4), are presented below.

3.6.3 Simulation for example problem

For the experimentation, the capacity of the service type 1 in SC as 500 is reduced in the step of 20. It was found that the optimal location solution did not change, but the optimal allocation result until the capacity was reduced to 420. Since the optimal location decision did not change, the associated cost remained constant. Because of the change in the allocation decision, the travel cost was found to change. The trend in the change is shown in Figure 3.6. A further reduction in the capacity changed the optimal location decisions as well. A reduction in the capacity is expected to increase the penalty cost. To avoid the severe increase in the penalty cost, the model tried to redistribute MTBs to other locations that were relatively far and had the capacity to absorb them. It has resulted in an increase in the travel cost. To visualize the effect of the capacity on the service level, the illustrative example problem was simulated with service type 1 in SC's capacity as 500 and 420, taking COV as 0.1, 0.2, 0.3, 0.4, and 0.5. The results are presented in Figure 3.7.

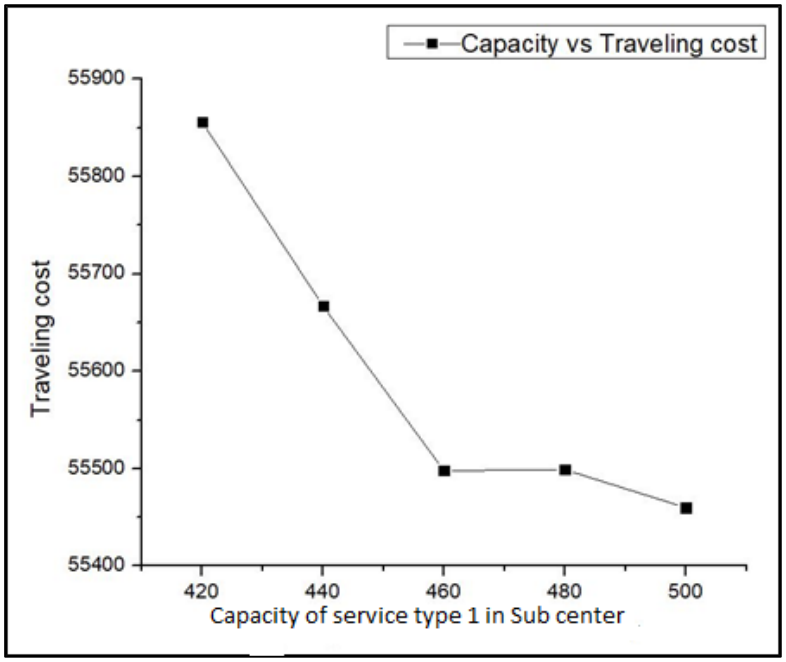

Figure 3.6: Effect of the capacity on overall traveling cost for the example problem

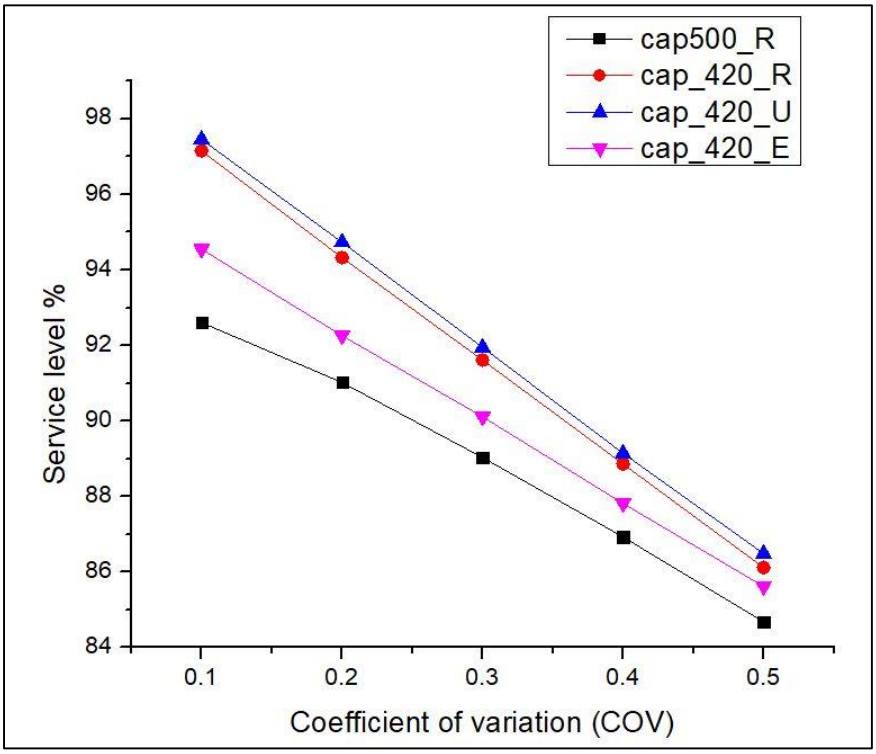

Figure 3.7:Effect of COV on the service level of BU's for the example problem

During the experimentation, the service level was found to decrease with the increase in the COV value. This validates the proposed model from the perspective of input-output

transformation. The decrease in the service level was found to be linear with the increase in the value of COV. In the example considered, the demand had no pattern across all the 18 locations. The existence of such relationship was further investigated for different demand patterns. Two different cases were taken. In the first case, the number of MTBs at each location was taken to be the same. In the other case, the number of MTBs at various locations was randomly fixed and was taken to follow a uniform distribution. In both the cases, however, the total number of MTBs for each service type was kept to be the same as for the example problem. It was done so for a meaningful comparison. The simulation results are shown in Figure 3.7 for all these cases. In this figure, cap500_R and cap420_R stand for the cases when the simulation was carried out with the demand data as for the example problem of Section 2.4 but with the respective capacity of service type 1 at SC as 500 and 420. cap420_U and cap420_E are for the cases when each SC capacity is 420, but the demand is respectively uniformly distributed or the same. The simulation results show that the service level decreases almost linearly with the increase in the value of COV irrespective of the capacity constraint or the nature of the demand in terms of its mean value. Similar results and observations were found in the cases of service type 2 and 3 in respective facility types. The simulation results tend to indicate that the service quality is highly dependent upon COV. To have more confidence in this finding, further investigations were carried out for many problem instances.

3.6.4 Simulation on randomly generated problems

For the purpose of computational experimentation, the problem instances generated in Chapter 2 are used here. Each problem instance was simulated for five values of COV as 0.1, 0.2, 0.3, 0.4, and 0.5 until the steady-state condition was reached. A total of 200 test instances were generated for the simulation purpose. The problem instances are solved by generating an initial

solution by the set covering model. The initial solution is then fed as a starting solution to the solver Gurobi 8.0.1. The optimization and simulation model is developed using Python interface on a personal computer equipped with Intel Core i7 3.40 GHz processor, 8GB RAM, and Microsoft Windows 10 Pro 64-bit operating system.

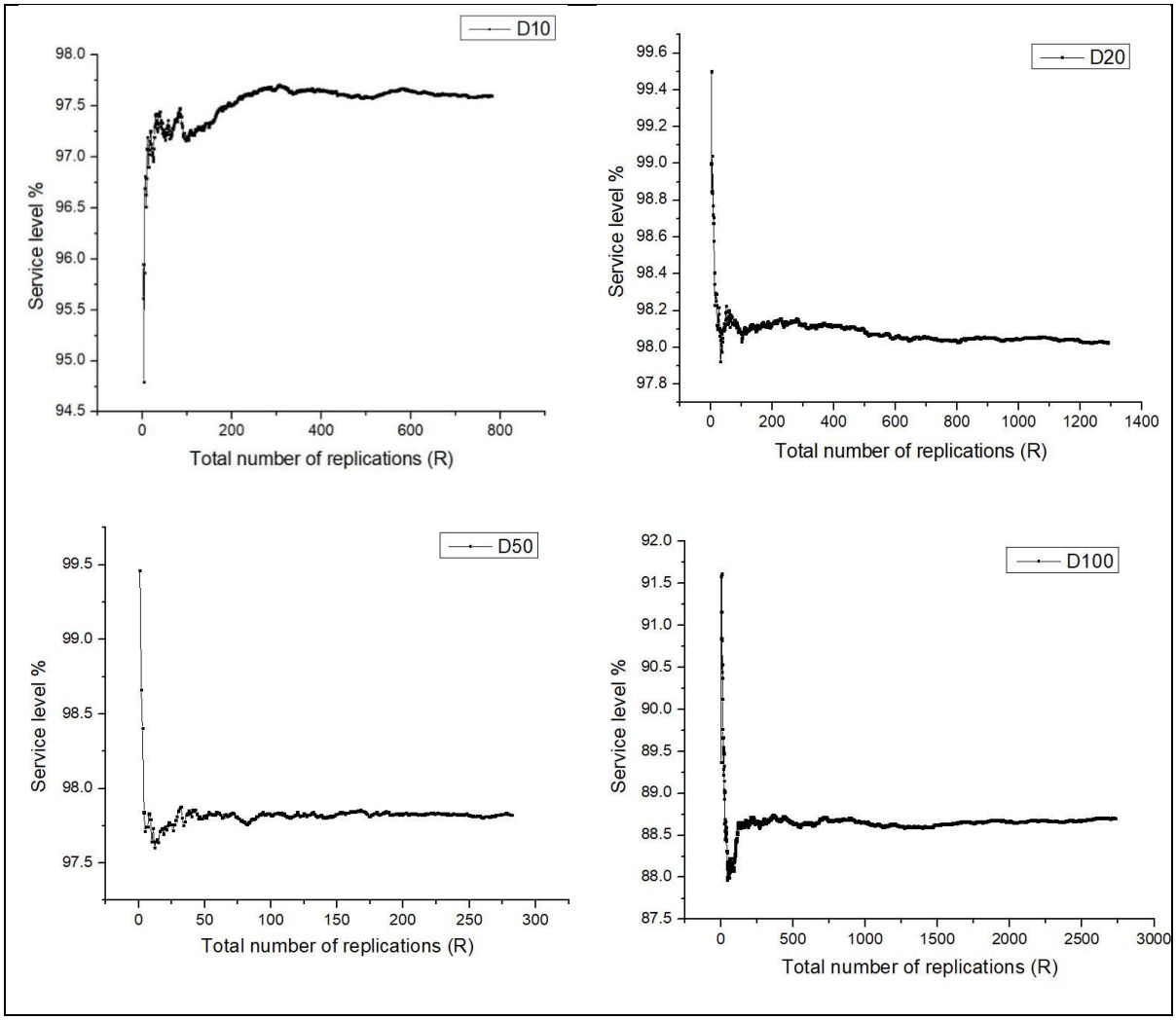

Figure 3.8: Variation in service level with the length of simulation run for COV as 0.5 for the problem class D10, D20, D50 and D100

The solutions obtained are fed into the simulation framework. Initially, simulation is carried out for ten independent replications and the same is further carried out for more number of replications to achieve error level of 0.1 in steady state condition. For COV value of 0.5, the

fluctuation in the service level with the length of the simulation run is shown in Figure 3.8 to illustrate the steady-state condition for a problem instance of each problem class. The detailed results are presented in Appendix A-1, and the summary of the simulation is presented here in Table 3.5.

In Table 3.5, a mean service level for an average of five problem instance of each problem class for each service type for varying COV is presented. Column 3 represents the total number of replications in the simulation. Columns 5, 8 and 11 show the mean service level of service types 1, 2, and 3, respectively. Columns 4, 7, and 10 show the lower limit of the confidence interval for service types 1, 2, and 3, respectively. Columns 6, 9, and 12 show the upper limit of the confidence interval for service types 1, 2, and 3, respectively.

From Table 3.5, it is observed that for all problem classes, the developed model is able to provide a service level of more than 95% for COV as 0.1 and 0.2 for service type 1. In other words, the number of facilities opened, to provide service type 1 have sufficient capacity to handle the demand variation expressed as 0.1 and 0.2 of COV for the 95% service level. For 0.3 and 0.4 values of COV, the service level for service type 1 is reported at around 93% and 90%, respectively, for all the problem classes. For the worst case, that is for 0.5 COV, the service level of service type 1 is around 88% for all problem classes. Service type 2 has a high service level because the capacity of service type 2 at a facility is more and is able to cater to the demand variation of 0.1 to 0.4 COV. The service level of service type 2 is more than 95% for 0.1, 0.2, 0.3, and 0.4 COV for all the problem classes; whereas around 93% service level is reported for service type 3 for all the problem classes. The service level of service type 3 is more than 95% in all the problem classes for each COV. The facilities opened have an underutilized capacity of service type 3. Due to that, enough capacity is available and thus the

model can provide a service level of more than 95% for each COV in each problem class. The reason behind the opening of a high number of facilities of type 3 and 2 is the considered coverage distance, which helps to handle the demand variation.

Table 3.5: Summary of Computational Experiments

Problem Class (1)	COV (2)	R (3)	SQ^1_{ll} (4)	SQ^1 (5)	SQ^1_{Ul} (6)	SQ^2_{ll} (7)	SQ^2 (8)	SQ^2_{ul} (9)	SQ^3_{ll} (10)	SQ^3 (11)	SQ^3_{ul} (12)
D10	0.1	1369	97.51	97.62	97.74	98.98	99.05	99.11	98.98	99.00	99.06
	0.2	8434	95.36	95.44	95.52	97.41	97.47	97.53	97.41	98.33	98.39
	0.3	11481	93.18	93.28	93.39	95.75	95.83	95.91	95.75	97.43	97.51
	0.4	10161	90.96	91.09	91.23	94.02	94.13	94.24	94.02	96.50	96.61
	0.5	32381	88.83	88.92	89.01	92.39	92.47	92.54	92.39	95.46	95.54
D20	0.1	851	97.89	97.98	98.08	99.06	99.13	99.20	99.06	100.00	100.00
	0.2	2366	95.47	95.58	95.70	97.89	97.98	98.07	97.89	100.00	100.00
	0.3	4079	92.86	92.99	93.12	96.58	96.68	96.78	96.58	99.94	100.00
	0.4	8057	90.38	90.50	90.62	95.20	95.30	95.40	95.20	99.80	99.90
	0.5	16407	87.77	87.87	87.98	93.69	93.77	93.86	93.69	99.58	99.67
D50	0.1	279	97.66	97.78	97.89	98.61	98.70	98.80	98.61	98.21	98.30
	0.2	1245	95.49	95.59	95.70	97.50	97.59	97.67	97.50	97.77	97.85
	0.3	2442	93.30	93.41	93.52	96.32	96.41	96.50	96.32	97.22	97.31
	0.4	4369	91.15	91.25	91.36	95.11	95.19	95.28	95.11	96.62	96.71
	0.5	6579	88.91	89.02	89.12	93.79	93.88	93.97	93.79	95.85	95.94
D100	0.1	196	97.46	97.57	97.67	98.35	98.42	98.50	98.35	98.37	98.44
	0.2	587	95.36	95.47	95.59	97.36	97.44	97.53	97.36	98.02	98.11
	0.3	1370	93.11	93.21	93.32	96.23	96.31	96.40	96.23	97.54	97.62
	0.4	2040	90.85	90.96	91.07	94.99	95.08	95.17	94.99	96.99	97.08
	0.5	3866	88.64	88.73	88.83	93.68	93.76	93.84	93.68	96.35	96.43

COV = Coefficient of variation

R = Total number of replications

SQ^1, SQ^2, SQ^3= Service level for service types 1, 2 and 3, respectively.

SQ^1_{ll}, SQ^2_{ll}, SQ^3_{ll} = Lower limit of confidence interval of service level for service types 1, 2 and 3, respectively.

SQ^1_{ul}, SQ^2_{ul}, SQ^3_{ul} = Upper limit of confidence interval of service level for service types 1, 2 and 3, respectively.

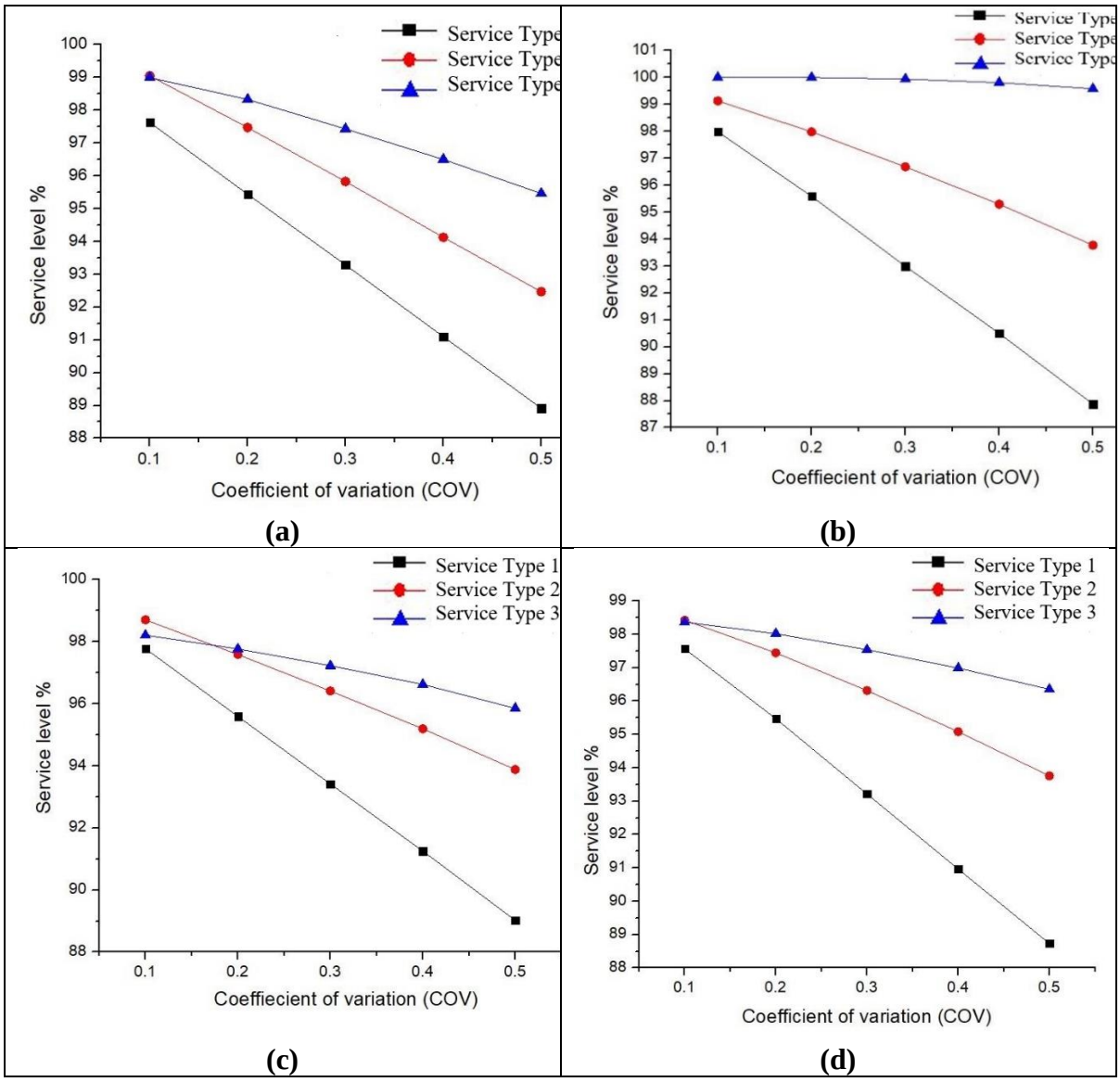

Figure 3.9: Service level vs COV for the problem instances (a) D10 (b) D20 (c) D50 (d) D100

3.4.5 Observations

The results from the simulation in terms of variation in the service level with COV value are shown in Figure 3.9. Figure 3.9 also shows that the service level decreases almost linearly with the increase in COV value. This figure also shows that the fall in the service level of service type 1 is more than that for the other service types. Similarly, the fall in service type 2 is more

in comparison to that for service type 3. It is because of the high level of the mean demand causing more spread or variation even for the same level of COV. It can be concluded that with the increase in the coefficient of variation, the service level deteriorates irrespective of problem class. The model is able to provide a service level of more than 95% for service types 1 till 0.2 COV. The developed model can handle the variation in demand up to 0.5 COV for service types 2 and 3 for a 95% service level. This simulation results show that the service level will be strongly governed by the mean level of the demand and the corresponding COV. These results also indicate that one should not restrict oneself to the location decisions obtained from the application of the HCFLP model but go with more facilities of each type. These observations are drawn from the 100 location problem.

3.7 Conclusions

This chapter is devoted to the service quality issue and proposes some metaheuristic approaches for solving large scale maternal healthcare facilities' location-allocation problems. The developed mathematical model ensures quality in service by considering a penalty on allocations of MTBs beyond the facility's limited capacity along with the availability and accessibility issues. The objective considered is the minimization of the total cost, which includes the cost of establishing a new facility, transportation costs and the penalty cost due to overburdening.

This chapter also proposed three metaheuristics, namely PSO, ABC and JAYA, to solve the large scale real-world problems. A major effort has gone into identifying the most effective and efficient metaheuristics out of the three in solving the model vis-à-vis the usual mathematical programming approach (use of Gurobi Solver). For this purpose, extensive computational experiments were carried out by solving 50 problem instances of varying sizes

generated randomly. Based on the statistical analyses, the PSO approach was found to be the most effective and efficient, even over the Gurobi Solver. The ABC approach was found to perform generally better than the JAYA approach in terms of solution quality and vice-versa in terms of CPU time used. The performance of the proposed metaheuristics has been found to be better than the usual optimizing approach adopted by Gurobi Solver. One may try to further improve the performance of these metaheuristics. It may be desirable to incorporate various search techniques such as local search, neighbourhood search and Tabu search.

The effect of randomly varying demand on the service quality was also examined using Monte-Carlo simulation method. The results of the simulation show that the service level decreases almost linearly with the increase in the coefficient of variation. This fall is more severe if the mean demand of a service type is large. The results of the simulation can be useful in deciding on additions to the capacity of various service types available at various locations in order to maintain the desired service level. In this way, the hybrid approach of combining simulation with the optimizing framework is expected to yield a much better practical solution. In India, the childbirth rate is almost constant. In case the same is not stationary in a country, the right planning framework would require the growth factor to be considered in the proposed formulation for effective planning.