

## CHAPTER

# MULTI-PERIOD MATERNAL HEALTHCARE FACILITY PLANNING CONSIDERING POPULATION GROWTH

## 4.1 Introduction

To respond to the increase in the demand with growth in the size of population with time, the capacity of the existing health facilities are generally increased as well as the same is upgraded to provide a higher level of services. It is quite common to find in India. With the growth in the population, the existing facilities get overburdened, and the quality of service deteriorates. Since proper healthcare is a priority of the governments in India, and particularly so for MTBs, as mentioned in Chapter 1, the quality of service should not be compromised beyond a point. To address this reality in the healthcare network planning, the mathematical models presented in earlier chapters cannot suffice as they did not consider the possibility of expanding and upgrading healthcare facilities to fulfil anticipated demand owing to population growth. This chapter addresses the strategic and tactical planning issues related to locating new facilities and upgrading various types of existing maternal healthcare facilities over a planning horizon. This problem is posed as a mixed-integer linear programming (MILP) model for the multi-period facility location-allocation model. The objective considered is the minimization of the overall cost, which includes the cost of establishment, cost of up-gradation, operating cost of the facilities, and transportation cost borne by mothers-to-be visiting these facilities. The model presented in this chapter is basically an extension of the HCFL model proposed in Chapter 2. Thus it is also computationally challenging to solve the model in real-time. To solve the problem described using the developed model in a computationally efficient manner, the

Benders decomposition algorithm is proposed along with several acceleration strategies such as valid inequalities, disaggregated Benders cuts, rolling horizon heuristic and parallelism to accelerate the performance. Besides, a heuristic involving the use of Benders decomposition philosophy is also proposed to solve considered problems efficiently. Particle swarm optimization metaheuristic along with local search is also proposed to obtain a good quality solution in much lesser time. Additionally, a hybridized fix-and optimize and simulated annealing approach is also proposed with various search space reduction techniques to obtain a good quality solution in a reasonable amount of CPU time.

The organization of the remaining part of the chapter is as follows. Section 4.2 describes the problem, and the mathematical formulation of the problem is presented in Section 4.3. The solution approaches are described in Section 4.4. The results of computational experiments are given in Section 4.5. Section 4.6 demonstrates the effectiveness of the proposed model by solving the maternal healthcare planning problem for the upcoming 15 to 20 years for the district of Chandauli in India. Finally, Section 4.7 concludes the chapter and presents the scope for future research work.

## **4.2 The Problem**

The maternal healthcare problem considered here has the same features as described in Chapters 2 and 3. Additionally, the issues resulting from population growth are also addressed. Mid and long-term strategic plans are required to open new healthcare facilities and upgrade and expand the existing healthcare facilities to handle the increased demand in future due to the number of MTBs growth. Therefore, it would require revisiting the network of maternal healthcare facilities to determine the location of the new healthcare facilities and up-gradation of the existing ones to serve the MTBs at different locations in various time periods of the

considered planning horizon. A facility type will keep providing the related type of services until it is upgraded. The problem considered here does not consider the possibility of expanding the capacity of the related service types that the facility type is meant to provide. It means that SC can be upgraded from PHC to CHC. CHC is the highest level of facility type, and the question of its up-gradation has been considered. It is assumed that the cumulative capacity for each service type over each facility type is much more than the demand of MTBs in any time period. Allocation of MTBs to a facility type has to be made such that it does not exceed the capacity available therein for a particular service type, or if so, only at a penalty.

Thus, the fixed cost incurred for establishing a facility at a location will depend upon its type and not the location. The same is also applicable to the up-gradation and operating cost of the facility. To meet the service demand of MTB in a given time period, sometimes up-gradation of the existing facility is more economical rather than opening a new one. The facility, once opened, cannot be closed, and thus its service will be available in future. The opened facility or existing facility at any given time period can be upgraded to a higher-level facility type in any subsequent time period to meet the demand of various service types. The up-gradation cost includes the cost incurred on the increase in the number of beds, staff, additional services, etc. For example, a lower-level facility type PHC can be upgraded to a higher-level facility type CHC by additionally providing neonatal services. The operating cost includes the doctor's salary, staff salary, maintenance cost, etc. The cost of construction, daily operating costs, and other expenses generally change with time and are mostly governed by the inflation rate prevailing in that period.

A transportation cost will be incurred by an MTB at location  $i$  to visit a facility at location  $j$ . It will generally be in proportion to the distance between locations  $i$  and  $j$ . Transportation cost is

also incurred during the referral visits. The transportation cost taken is constant over the whole planning horizon. The objective is to minimize the sum of the costs on establishing the facilities, up-gradation of the facilities, operating cost of the facilities, cost incurred on visiting these facilities, including referrals, and penalty costs. As the facilities and the services provided by them are nested, the problem is referred to as a Multi-Period Hierarchical Capacitated Facility Location-Allocation (MHCFL) problem in subsequent elaborations and discussions.

### 4.3 Model Formulation

Most of the expressions used in the formulation are the same as were developed in Chapters 2 and 3. However, the same is rewritten here for ready reference and continuity.

#### Sets and Indices

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

$J_e$  : set of locations where facilities are already existing, indexed by  $j_e$

$J_n$  : set of potential locations where new facilities can be established, indexed by  $j_n$

$J$  : set of potential locations of facilities,  $J = J_e \cup J_n$ , indexed by  $j, k$  and  $a$

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

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

$T$  : set of the time periods,  $T = \{1, 2, \dots, c\}$ , indexed by  $t$  and  $\tau$

#### Parameters

$M$  : a big number

$P$  : penalty cost per additional MTB beyond the capacity of any service type at any 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  $j \in J$  (used for locations of MTBs)

$d_{jk}$  : distance between locations  $j \in J$  and  $k \in J$  (used for locations of facilities)

$C_{ij}$  : 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 the location  $k \in J$

$F_j^{ft}$  : fixed cost on establishing a facility of type  $f \in F$  at the location  $j \in J_n$  in a time period  $t \in T$

$O_j^{ft}$  : operating cost of a facility of type  $f \in F$  at the location  $j \in J$  during a time period  $t \in T$

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

$U_j^{fgt}$  : cost to upgrade the facility of type  $f \in F$  to facility of type  $g \in F$  at the location  $j \in J$  during a time period  $t \in T$  where  $g > f$

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

$\kappa_i^t$  : fraction showing growth in number of MTBs at a location  $i \in I$  during a time period  $t \in T$

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

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

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

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

### Decision variables

$x_{ij}^{lt}$ : number of MTBs at location  $i \in I$  allocated to a facility at a location  $j \in J$  to receive service type  $l \in L$  during a time period  $t \in T$

$x_{jk}^{lmt}$ : number of MTBs seeking service type  $l \in L$  at a facility  $j \in J$  and which are referred to a facility  $k \in J$  for a higher service type  $m \in L$  during a time period  $t \in T$

$\bar{x}_j^{lt}$ : an excess number of MTBs allocated to a facility at a location  $j \in J$  to receive service type  $l \in L$  during a time period  $t \in T$

$y_j^{ft} = \begin{cases} 1, & \text{if a new facility of type } f \in F \text{ is located in } j \in J_n \text{ during time period } t \in T, \\ 0, & \text{otherwise.} \end{cases}$

$z_j^{ft} = \begin{cases} 1, & \text{if a facility of type } f \in F \text{ is operating at a location } j \in J \text{ during time period } t \in T, \\ 0, & \text{otherwise.} \end{cases}$

$\bar{y}_j^{fgt} = \begin{cases} 1, & \text{if a facility of type } f \in F \text{ is upgraded to facility of type } g \in F (g > f) \text{ at a location } j \in J \\ & \text{during timeperiod } t \in T, \\ 0, & \text{otherwise.} \end{cases}$

Using the above notations, the mathematical model of the Multi-Period Hierarchical Capacitated Facility Location-Allocation (MHCFL) problem is presented hereunder.

$$\text{Minimize } \left\{ \begin{array}{l} \sum_{j \in J_n} \sum_{t \in T} \sum_{f \in F} F_j^{ft} y_j^{ft} + \sum_{j \in J} \sum_{t \in T} \sum_{f \in F} U_j^{fgt} \bar{y}_j^{fgt} + \sum_{j \in J} \sum_{t \in T} \sum_{f \in F} O_j^{ft} z_j^{ft} \\ + \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} \sum_{t \in T} c_{ij} x_{ij}^{lt} + \sum_{k \in J} \sum_{j \in J} \sum_{m \in L} \sum_{l \in L} \sum_{t \in T} c_{jk} x_{jk}^{lmt} \\ + \sum_{i \in I} \sum_{l \in L} \sum_{t \in T} P \bar{x}_j^{lt} \end{array} \right\} \quad (4.1)$$

## Subject to

### Flow constraints

$$\sum_{j \in J} x_{ij}^{lt} = (1 + \kappa_i^l) W_i^l, \quad \forall i \in I, \forall l \in L, \forall t \in T \quad (4.2)$$

$$\sum_{k \in J} x_{jk}^{lmt} = \theta^{lm} \left\{ \sum_{i \in I} x_{ij}^{lt} + \sum_{a \in J} \sum_{p \in L} x_{aj}^{plt} \right\}, \quad \forall j \in J, \forall l \in L, \forall m \in L, \forall t \in T, m > l > p \quad (4.3)$$

### Capacity constraints

$$\sum_{i \in I} x_{ij}^{lt} + \sum_{k \in J} \sum_{p \in L} x_{kj}^{plt} - \bar{x}_j^{lt} \leq \sum_{f \in F} Q^{lf} z_j^{ft}, \quad \forall j \in J, \forall t \in T, \forall l \in L, p < l \quad (4.4)$$

### Constraints related to coverage distance

$$x_{ij}^{lt} \leq \sum_{f \in F} \alpha_{ij} z_j^{ft} M^{lf}, \quad \forall i \in I, \forall j \in J, \forall l \in L, t \in T \quad (4.5)$$

$$x_{jk}^{lmt} \leq \sum_{f \in F} \beta_{jk} z_k^{ft} M^{mf}, \quad \forall j \in J, \forall k \in J, \forall l \in L, \forall m \in L, \forall t \in T, m > l \quad (4.6)$$

### Constraints related to establishing a facility at new locations

$$z_j^{ft} = y_j^{ft} \quad t = 1, \forall j \in J_n, \forall f \in F \quad (4.7)$$

$$z_j^{ft} \leq z_j^{f(t-1)} + y_j^{ft} + \sum_{g \in F} \bar{y}_j^{fgt} \quad t \geq 2, \forall j \in J_n, \forall f \in F, f > g \quad (4.8)$$

$$\sum_{f \in F} z_j^{ft} \geq \sum_{\tau \in T, \tau \leq t} \sum_{f \in F} y_j^{f\tau} \quad t \geq 2, \forall j \in J_n, \forall t \in T \quad (4.9)$$

$$\sum_{g \in F} \bar{y}_j^{fgt} \leq z_j^{f(t-1)} \quad t \geq 2, \forall j \in J_n, f \in F \quad (4.10)$$

$$y_j^{ft} + \sum_{g \in F} \bar{y}_j^{fgt} \leq 1 \quad \forall j \in J_n, \forall f \in F, t \in T \quad (4.11)$$

$$\sum_{f \in F} z_j^{ft} \leq 1 \quad \forall j \in J_n, \forall t \in T \quad (4.12)$$

$$\sum_{t \in T} \sum_{f \in F} y_j^{ft} \leq 1 \quad \forall j \in J_n \quad (4.13)$$

**Constraints related to existing facilities**

$$\sum_{f \in F} z_j^{ft} = 1 \quad \forall j \in J_e, \forall t \in T \quad (4.14)$$

$$\sum_{g \in F} \bar{y}_j^{fgt} \leq k_j^f \quad t = 1, \forall j \in J_e, \forall f \in F, g > f \quad (4.15)$$

$$z_j^{ft} \leq k_j^f + \sum_{g \in F} \bar{y}_j^{gft} \quad t = 1, \forall j \in J_e, \forall f \in F, g < f \quad (4.16)$$

$$z_j^{ft} = z_j^{f(t-1)} + \sum_{g \in F} \bar{y}_j^{fgt} \quad t \geq 2, \forall j \in J_e, \forall f \in F, g > f \quad (4.17)$$

**Integrality and other constraints**

$$Q^{2I} = Q^{3I} = Q^{3H} = 0 \quad (4.18)$$

$$\bar{y}_j^{fgt} = 0 \quad \forall j \in J, f \in F, g \in F, f \geq g, t \in T \quad (4.19)$$

$$x_{ij}^{lt}, x_{jk}^{mt} \geq 0, \quad \forall i \in I, \forall j \in J, \forall k \in J, \forall l \in L, \forall m \in L, \forall t \in T \quad (4.20)$$

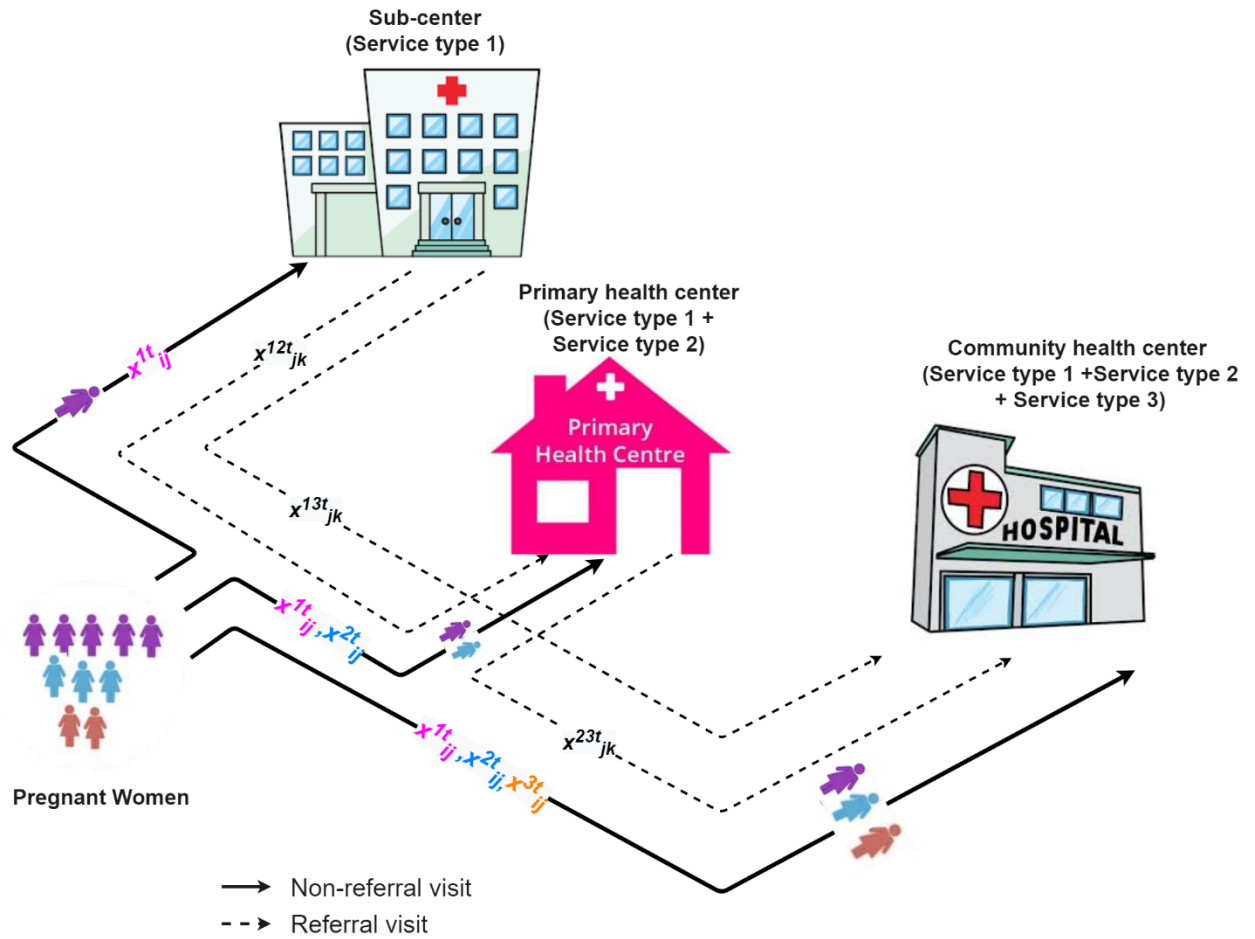
$$y_j^{ft}, z_j^{ft}, \bar{y}_j^{fgt} \in \{0, 1\}, \quad \forall j \in J, \forall f \in F, \forall g \in F, \forall t \in T \quad (4.21)$$

Figure 4.1 shows the flows of MTBs to various facility types from a location. The mathematical model can be understood from this. The objective function (4.1) shows the minimization of the total cost over the planning horizon. Various terms in the objective function expression respectively represent the fixed cost of establishment, up-gradation cost, operating cost of the healthcare facility, cost on travel for visiting a facility, cost on travel upon referral, and the penalty cost, on overall basis. Constraint (4.2) ensures complete fulfillment of the demand of various service types of all the MTBs in each time period. The number of MTBs referred from lower to higher level service types is determined by Constraint (4.3). Constraint (4.4) is a capacity constraint that determines excess number of MTBs allocated to each facility type for the realted service types in each time period. From this constraint and the last term of the



objective function expression (4.1), it is obvious that the model will not permit any allocation beyond the capacity if the penalty cost ( $P$ ) is made very high. The proposed formulation assumes that the up-gradation and/or addition of a facility will increase the availability in the same time period of the planning horizon. Constraint (4.5) states that the MTBs at a location  $i \in I$  can only be assigned to a facility at location  $j \in J$  if the required service type is available in that facility and the same is available within the coverage distance. Constraint (4.6) will ensure the referral of MTBs for a higher level of service type to a facility within its permissible maximum coverage distance. Constraints (4.7) to (4.13) are concerned with establishing new facilities or up-gradating of facilities in a period of the planning horizon. Upgradation is applicable for all the facilities irrespective of when these were established or upgraded. Constraint (4.7) takes cognizance of the operation of a facility from a location even if it is newly established at that location in the first period itself. Constraint (4.8) ensures the continuation of a facility in subsequent periods, whether it is newly established or upgraded to provide a higher-level service type. Constraint (4.9) is to ensure the continuity of a facility in providing the related services. Constraint (4.10) ensures the possibility of up-gradation of a facility to a higher level only when it was operating in the earlier time period. Constraint (4.11) restricts the opening and up-gradation of a facility in the same time period. Constraints (4.12) and (4.13) impose the restriction that only a single facility can operate at a location at any time period, whether existing, upgraded, or newly located. Constraint (4.14) will ensure the working of only one facility type at any location at any point in time. Constraints (4.15) to (4.17) are concerned with the up-gradation of an existing facility. Constraint (4.15) permits only the up-gradation of an existing facility type at the beginning of the planning horizon. Constraint (4.16) states that a facility type  $f$  will be operating in the first time period only if exist or upgraded to

facility type  $f$ . permits the up-gradation of a facility at a location in the first time period provided it exists. Upgradation in subsequent periods is taken care of by constraint (4.17). Upgradation of SC (represented by index  $I$ ) to PHC (represented by index  $II$ ) or CHC (represented by index  $III$ ), or PHC to CHC may bring change in the capacity of these facilities for various service types. Constraint (4.18) takes care of the factual nature of the various facility types. SCs cannot provide services of types 1 and 2. Similarly, PHCs cannot provide service type 3. Constraints (4.19) to (4.21) express the nature of decision variables. In fact, the number of allocations  $x_{ij}^{In}$  and  $x_{jk}^{Im}$  should be an integer as the number of MTBs cannot be fractional. The number of referrals taken as a proportion of the MTBs allocated to the healthcare facilities is given by the constraints (4.2) and (4.3). As a result, the resulting number may be fractional. This will ask for equating some of the integer numbers to be equal to a real number, and the same is impossible. To avoid this difficulty, allocation variables are being treated as continuous variables. Since numbers defining the corresponding allocations are reasonably high in the Indian context, even rounding off non-integer values to the nearest value will not practically create much of a problem. Besides, treating these variables as non-integer variables will help in reducing the computational complexity of the problem.



**Figure 4.1: Allocation of mothers-to-be to different types of facility**

## 4.4 Solution Approaches

MHCFL formulation was coded in Python 3.8 (Rossum & Drake, 2011) to solve the related problems using Gurobi 9.0.3, which is the state of the art mathematical programming solver. The solver could provide optimal solutions for small and some medium-size problem instances within a reasonable time. However, the numerical experiments revealed that the computational time increases exponentially as the problem size increases. The solver could not even produce a feasible solution for many large-size instances. Therefore, obtaining an optimal or a good quality solution in a reasonable amount of time is a challenge for the MHCFL problem. To

address this issue, various approaches are proposed to solve the problem considered in this chapter efficiently and effectively. The same are elaborated below.

## **4.5 Benders decomposition**

Benders Decomposition (BD) algorithm is a widely used exact solution approach for solving large-scale combinatorial optimization problems pioneered by Benders (2005). It takes advantage of the problem's structure by partitioning it into a master problem (MP) and a sub-problem (SP). MP taken is a relaxed version of the original problem with complicating variables (binary variables - as in our case) and associated constraints. SP framed has to deal with continuous variables of the original problem and the remaining constraints while taking the values of binary variables from the current solution of MP. MP and the SP are solved iteratively by providing solution from each other until the optimal solution to the problem resulted where going through the iteration no change in the solution of MP or SP is observed. If the solution of MP is feasible to the SP, an optimality cut is generated to possibly improve the solution of MP in consecutive iterations. Otherwise, a feasibility cut is generated and fed back to the MP to attain a feasible solution for MP. Thus, the MP's solution provides input in the form of values of binary variables to the SP, while the solution of SP is used to provide feedback to the MP. Instead of solving the SP directly, its dual problem which is named as dual sub-problem (DSP), is solved for the computational advantage. In the present case of the minimization problem, the MP gives a lower bound (LB), while the solution of DSP along with the objective value of MP provides an upper bound (UB) to the original problem. Problem decomposition and iterative solution strategy of BD reduces the computational burden significantly making BD a lot more acceptable. BD found its successful application in many areas such as scheduling (Canto, 2008), facility location (Boland et al. 2016), health care

(Heching et al. 2019; Karamyar et al. 2018), telecommunications (Kewcharoenwong and Üster, 2014), network design (Esmailbeigi et al. 2021; Gong and Zhang, 2022; Reddy et al. 2022; Cheng, 2022) and chemical process design (Zhu and Kuno, 2003) to name a few. The work of Rahmaniani et al. (2017) can be referred to for a thorough review of the BD algorithm and applications.

MHCFL model, presented in Section 4.3, can be decomposed into location and allocation problems. The location problem deals with binary variables related to location ( $y_j^{ft}$ ), upgradation ( $\bar{y}_j^{fgt}$ ) and operation ( $z_j^{ft}$ ) of a facility only, the variables responsible for the major computational burden. The allocation problem would require the value of  $z_j^{ft}$ . Since MP has resulted the values for  $z_j^{ft}$  variable, the same can be taken as such by the allocation problem which is a linear programming problem and thus easy to solve. This special structure of the MHCFL problem is the main motivational reason to implement and use the BD algorithm. Consequently, the location problem is viewed as an MP, and the allocation problem as an SP. The formulation of MP is as follows.

### Master Problem (MP)

$$\text{Minimize } \left\{ \sum_{j \in J_n} \sum_{t \in T} \sum_{f \in F} F_j^{ft} y_j^{ft} + \sum_{j \in J} \sum_{t \in T} \sum_{f \in F} U_j^{fgt} \bar{y}_j^{fgt} + \sum_{j \in J} \sum_{t \in T} \sum_{f \in F} O_j^{ft} z_j^{ft} + \sum_{t \in T} \xi^t \right\} \quad (4.22)$$

Subject to constraints (4.7) to (4.19) and (4.21) of **MHCFL**.

### Feasibility cut:

$$\begin{aligned} & \sum_{i \in I} \sum_{l \in J} \sum_{t \in T} (1 + \kappa_i^t) W_i^l \lambda_i^{2,lt} + \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \sum_{t \in T} Q^{lf} z_j^{ft} \lambda_j^{4,lt} \\ & + \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \sum_{t \in T} \alpha_{ij} z_j^{ft} M^{lf} \lambda_{ij}^{5,lt} + \sum_{i \in I} \sum_{j, k \in J} \sum_{l, m \in L} \sum_{f \in F} \sum_{t \in T} \beta_{jk} z_k^{ft} M^{mf} \lambda_{jk}^{6,lm} \leq 0 \end{aligned} \quad (4.23)$$

**Optimality cut:**

$$\begin{aligned}
& \sum_{i \in I} \sum_{l \in L} \sum_{t \in T} (1 + \kappa_i^l) W_i^l \lambda_i^{2,lt} + \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \sum_{t \in T} Q^{lf} z_j^{ft} \lambda_j^{4,lt} \\
& + \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \sum_{t \in T} \alpha_{ij} z_j^{ft} M^{lf} \lambda_{ij}^{5,lt} + \sum_{i \in I} \sum_{j, k \in J} \sum_{l, m \in L} \sum_{f \in F} \sum_{t \in T} \beta_{jk} z_k^{ft} M^{mf} \lambda_{jk}^{6,lm} \leq \xi^t
\end{aligned} \tag{4.24}$$

The terms in the objective function (4.22) of the MP are in line with the objective function (4.1) of the MHCFL model except for a new auxiliary variable  $\xi^t$  ( $\xi^t \geq 0$ ), which takes continuous variables terms of the objective function (4.1) into account. Constraints (4.7) to (4.19) and (4.21) are from the MHCFL model that relates the binary variables in the model. Constraints (4.23) and (4.24) are the feasibility and optimality cuts, respectively. These are Benders cuts, are devised from the solution of the DSP. Here  $\lambda_i^{2,lt}$ ,  $\lambda_j^{3,lm}$ ,  $\lambda_j^{4,lt}$ ,  $\lambda_{ij}^{5,lt}$ , and  $\lambda_{jk}^{6,lm}$  are the dual variables in DSP and  $\lambda_i^{2,lt}$ ,  $\lambda_j^{3,lm}$ ,  $\lambda_j^{4,lt}$ ,  $\lambda_{ij}^{5,lt}$ , and  $\lambda_{jk}^{6,lm}$  represent the values of these dual variables obtained by solving the DSP. Formulation of SP and the corresponding DSP are as follows.

**Sub-Problem (SP)**

$$\text{Minimize } \left\{ \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} \sum_{t \in T} c_{ij} x_{ij}^{lt} + \sum_{k \in J} \sum_{j \in J} \sum_{m \in L} \sum_{l \in L} \sum_{t \in T} c_{jk} x_{jk}^{lmt} + \sum_{i \in I} \sum_{l \in L} \sum_{t \in T} P \bar{x}_j^{lt} \right\} \tag{4.25}$$

**Subject to**

$$\sum_{j \in J} x_{ij}^{lt} = (1 + \kappa_i^l) W_i^l, \quad \forall i \in I, \forall l \in L, \forall t \in T \tag{4.26}$$

$$\sum_{k \in J} x_{jk}^{lmt} = \theta^{lm} \left\{ \sum_{i \in I} x_{ij}^{lt} + \sum_{a \in J} \sum_{p \in L} x_{aj}^{plt} \right\}, \quad \forall j \in J, \forall l \in L, \forall m \in L, \forall t \in T, m > l > p \tag{4.27}$$

$$\sum_{i \in I} x_{ij}^{lt} + \sum_{k \in J} \sum_{p \in L} x_{kj}^{plt} - \bar{x}_j^{lt} \leq \sum_{f \in F} Q^{lf} \bar{z}_j^{ft}, \quad \forall j \in J, \forall t \in T, \forall l \in L, p < l \quad (4.28)$$

$$x_{ij}^{lt} \leq \sum_{f \in F} \alpha_{ij} \bar{z}_j^{ft} M^{lf}, \quad \forall i \in I, \forall j \in J, \forall l \in L, t \in T \quad (4.29)$$

$$x_{jk}^{lmt} \leq \sum_{f \in F} \beta_{jk} \bar{z}_k^{ft} M^{mf}, \quad \forall j \in J, \forall k \in J, \forall l \in L, \forall m \in L, \forall t \in T, m > l \quad (4.30)$$

The objective function (4.25) of the SP contains allocation and penalty variables. Constraints (4.26) and (4.27) are reproduced from the MHCFL model, and they are the same as constraints (4.2) and (4.3), respectively. Constraints (4.28), (4.29) and (4.30) deal with the fixed value of binary variables. To quickly obtain a better UB, a DSP has to be solved. Considering the dual variables  $\lambda_i^{2,lt}$ ,  $\lambda_j^{3,lm}$ ,  $\lambda_j^{4,lt}$ ,  $\lambda_{ij}^{5,lt}$ , and  $\lambda_{jk}^{6,lm}$  associated with respective constraints from (4.26) to (4.30), the dual formulation of sub problem is stated below.

### Dual Sub problem (DSP)

$$\text{Maximize } \left\{ \begin{aligned} & \sum_{i \in I} \sum_{l \in L} \sum_{t \in T} (1 + \kappa_i^l) W_i^l \lambda_i^{2,lt} + \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \sum_{t \in T} Q^{lf} \bar{z}_j^{ft} \lambda_j^{4,lt} \\ & + \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \sum_{t \in T} \alpha_{ij} \bar{z}_j^{ft} M^{lf} \lambda_{ij}^{5,lt} + \sum_{i \in I} \sum_{j, k \in J} \sum_{l, m \in L} \sum_{f \in F} \sum_{t \in T} \beta_{jk} \bar{z}_k^{ft} M^{mf} \lambda_{jk}^{6,lm} \end{aligned} \right\} \quad (4.31)$$

$$\lambda_i^{2,lt} - \theta^{lm} \sum_{m \in L} \lambda_j^{3,lm} + \lambda_j^{4,lt} + \lambda_{ij}^{5,lt} \leq C_{ij}, \quad \forall i \in I, \forall j \in J, \forall l \in L, \forall t \in T, m > l \quad (4.32)$$

$$\lambda_i^{3,lm} - \theta^{lm} \lambda_j^{4,lm} + \lambda_j^{5,lt} + \lambda_{jk}^{6,lm} \leq C_{jk}, \quad \forall i \in I, \forall j \in J, \forall l \in L, \forall m \in L, \forall t \in T, m > l \quad (4.33)$$

$$\lambda_j^{4,lt} \leq P, \quad \forall j \in J, \forall l \in L, \forall t \in T \quad (4.34)$$

$$\lambda_i^{2,lt}, \lambda_j^{3,lm} \text{ are URS} \quad (4.35)$$

$$\lambda_j^{4,lt}, \lambda_{ij}^{5,lt}, \lambda_{jk}^{6,lm} \leq 0 \quad (4.36)$$

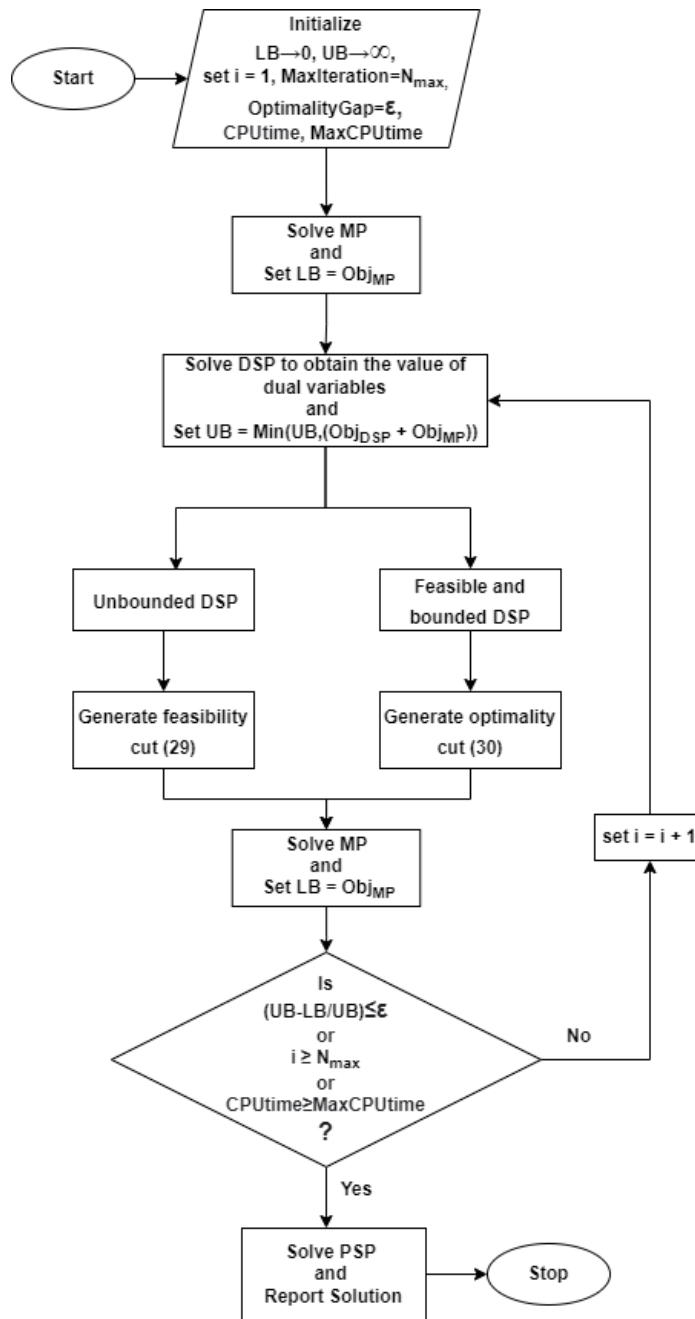
The objective function of DSP (4.31) is having a sense of maximization since SP is of a minimization nature. Constraints (4.32), (4.33) and (4.34) are the dual constraints related to continuous variables  $x_{ij}^{lt}$ ,  $x_{jk}^{lm}$  and  $\bar{x}_j^{lt}$ , respectively. Constraints (4.35) and (4.36) show the nature of the dual variables.

The above discussed MP and DSP are solved iteratively to obtain the final solution. Many different strategies can be used to faster the iteration and improve the bounds. In the following discussion, first, we discuss classical Benders decomposition. Then by adding various acceleration strategies, accelerated Benders decomposition is explained. Lastly, the Benders type heuristic is developed to quickly solve the MP.

#### **4.5.1 Classical Benders decomposition**

The schematic representation of the Classical BDA is shown in Figure 4.2. In the first iteration of classical BD, the MP is solved without any Benders cuts. The solution obtained through MP is then used to solve the DSP. The objective function value of DSP and MP (except the value of an auxiliary variable) is added to calculate the upper bound (UB). UB will be updated only when it has improved value while using the of a binary variable  $\bar{z}_j^{ft}$  obtained from MP. If the DSP results in an unbounded solution (i.e., infeasible SP), then a feasibility cut (4.23) will be generated. Otherwise, an optimality cut (4.24) is generated if the DSP has a bounded solution (i.e. feasible SP). These cuts will be inserted into the MP in each iteration.





**Figure 4.2: Framework of classical Benders decomposition**

The algorithm terminates when the required convergence or optimality gap is achieved, or it touches a specified maximum number of iterations or maximum CPU time limit, whichever is touched earlier. The final solution can be obtained by solving the SP using the solution of the MP.

## 4.5.2 Accelerated Benders decomposition

The classical BD algorithm requires a large number of iterations and is difficult to converge even for moderate size problem instances of MHCFL. As the size of the problem increases, it becomes almost impossible to have convergence within the specified CPU time limit. This can be attributed to the weaker Benders cuts, a lack of appropriate initial solution, and the size and complexity of the MP and DSP. To get over these issues, the strategies adopted to accelerate the convergence of the BD algorithm are detailed in the following sub-sections.

### 4.5.2.1 Additional Valid Inequalities

Valid inequalities added to the MP help to remove or cut off the non-integer solutions from the solution space. Valid inequalities are expected to provide a better lower bound in the case of the minimization problem (Pochet & Wolsey, 2006). For this purpose, the following additional valid inequalities are proposed.

**Valid Inequality I.** At a given time period  $t \in T$ , there should be at least one operating facility available within the coverage distance of a location  $i \in I$ . It is expressed as:

$$\sum_{j \in J} \sum_{f \in F} \alpha_{ij} z_j^{ft} \geq 1, \quad \forall i \in I, \forall t \in T. \quad (4.37)$$

The above valid inequality is similar to the set covering constraint and can be proved following the description of the problem given in Daskin (2011).

**Valid Inequality II.** For a referral from each lower-level facility at a location  $j \in J$  in time period  $t \in T$ , there must be a higher-level facility available at a location  $k \in J$  within the pre-specified coverage distance from  $j \in J$  and in time period  $t \in T$ .

$$z_j^{ft} \leq \sum_{k \in J} \sum_{g \in F} \beta_{jk} z_k^{gt}, \quad \forall j \in J, \forall f \in F, \forall t \in T, f < g \quad (4.38)$$

Constraint (4.6) of the MHCFL model imposes a requirement that MTB which is getting a service level  $l \in L$  at a facility  $j \in J$  in time period  $t \in T$  can only be referred to a facility at location  $k \in J$  for a higher level service  $m \in L$ , only if  $\beta_{jk} = 1$ . Now, if  $\sum_{k \in J} \sum_{g \in f, g > f} \beta_{jk} z_k^{gt} = 0$  for an operating facility at the location  $j \in J$  in time period  $t \in T$  and  $x_{jk}^{lm} > 0$ , it would violate Constraint (4.6). If there is a referral from a lower-level facility, i.e.,  $z_j^{ft} = 1$ , then, at least one higher-level facility should be available to fulfil that demand of referral, i.e.,  $z_k^{gt} = 1$  within coverage distance and in a given time period  $t \in T$ , where  $f \in F$  denotes lower-level facility and  $g \in F$  is for the higher-level facility.

**Valid Inequality III.** In any time period  $t \in T$ , the cumulative capacity of the operating facilities providing service level  $l \in L$  within the coverage distance from the location  $i \in I$  must be greater than or equal to  $W_i^l$ . It is expressed as:

$$\sum_{j \in J} \sum_{f \in F} \alpha_{ij} z_j^{ft} M^f \geq (1 + \kappa_i^l) W_i^l, \quad \forall i \in I, \forall l \in L, \forall t \in T \quad (4.39)$$

Constraint (4.5) of the MHCFL model ensures the allocation of MTBs from location  $i \in I$  seeking service level  $l \in L$  to the appropriate type of facilities, which are operating within a pre-specified coverage distance from the location  $i \in I$  in a given time period  $t \in T$ . Further, Constraint (4.2) demands that all MTBs at a location  $i \in I$  in a time period  $t$  who are seeking service level  $l \in L$  need to be allocated to an appropriate operating facility. Therefore, if there exists a non-zero population of MTBs at a location  $i \in I$  and if Constraints (4.2) and (4.6) hold good, the total availability in the operating facilities within the coverage distance from  $i$  must at least be equal to the number of MTBs seeking service level  $l \in L$  in time period  $t \in T$ .

The valid inequalities I, II and III are found to improve the LB and thus overall computational efficiency.

#### **4.5.2.2 Rolling-horizon heuristic for the initial solution of MP**

The upper bound obtained in initial iterations of BD is observed to be poor. It may cause the algorithm to explore inferior parts of the feasible region (Santoso et al. 2005). If BD algorithm is provided with a good initial feasible solution (with a good initial upper bound), the algorithm eventually will lead to a better quality of Benders cuts.

In this work, a rolling-horizon heuristic is proposed to find a reasonably good quality feasible solution quickly. In this heuristic, the MHCFL model is solved for each individual period of the planning horizon sequentially while considering the existing healthcare facilities in the immediately preceding period. This decomposition is possible because the facilities, either planned to be established or upgraded to a higher level, can come up during the period and start providing the related services in the same period itself. In the MHCFL model, the allocation decision is mainly governed by Constraints (4.2) to (4.6) are to be applied for the corresponding time period. The decision related to the allocation of MTBs in a given time period is taken to be independent of the allocation decision of the previous time period. The decision of establishing new facilities and up-grading existing ones depends upon the facilities operating in the previous time period. Or using this approach, the formulation named as 'P' and shown below is based on each time period.

Objective function (4.1)

Subject to Constraints (4.2) to (4.6), (4.11) to (4.14) and (4.18) to (4.21) of MHCFL

$$z_j^f = y_j^f + \sum_{g \in F} \bar{y}_j^{fg} \quad \forall j \in J_n, \forall f \in F, f > g \quad (4.40)$$

$$\sum_{g \in F} \bar{y}_j^{fg} \leq 1 \quad \forall j \in J_n, f \in F, f > g \quad (4.41)$$

In 'P', the objective function and the remaining constraints are rewritten for the respective time period. Constraints (4.40) and (4.41) are analogous to Constraints (4.8) and (4.10) of the MHCFL model. Each 'P' is then solved sequentially period wise and the set of the existing facility is updated simultaneously. The final solution to the original MHCFL will be obtained by integrating the solutions of 'P' for all the periods of the planning horizon. This sequential addressing of the problem requires less computational effort due to the reduced number of constraints and variables. This consideration helps in obtaining a reasonably good upper bound for the MHCFL problem quickly. The solution obtained by this rolling horizon heuristic is used as an initial solution for the DSP in the first iteration of the BD algorithm.

#### 4.5.2.3 Disaggregation of Benders cuts

As the Benders algorithm progresses, the size of MP increases due to the cuts added in each iteration. The disaggregation of Bender's cut may result in a better convergence at the expense of a larger MP (Santoso et al., 2005). The feasibility and optimality cuts added into the MP in each iteration of BD can be disaggregated for each time period  $t \in T$ . The modified feasibility, expressed by inequality (4.42), and optimality cuts, represented by inequality (4.43), replace respective feasibility (4.23) and optimality cuts (4.24).

#### Disaggregated feasibility cut

$$\begin{aligned} & \sum_{i \in I} \sum_{l \in L} (1 + \kappa_i^l) W_i^l \lambda_i^{2,lt} + \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} Q^{lf} z_j^{ft} \lambda_j^{4,lt} \\ & + \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \alpha_{ij} z_j^{ft} M^{lf} \lambda_{ij}^{5,lt} + \sum_{i \in I} \sum_{j,k \in J} \sum_{l,m \in L} \sum_{f \in F} \beta_{jk} z_k^{ft} M^{mf} \lambda_{jk}^{6,lm} \leq 0 \quad \forall t \in T \end{aligned} \quad (4.42)$$

### Disaggregated optimality cut

$$\begin{aligned}
& \sum_{i \in I} \sum_{l \in L} (1 + \kappa_i^l) W_i^l \lambda_i^{2,lt} + \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} Q^{lf} z_j^{ft} \lambda_j^{4,lt} \\
& + \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \alpha_{ij} z_j^{ft} M^{lf} \lambda_{ij}^{5,lt} + \sum_{i \in I} \sum_{j,k \in J} \sum_{l,m \in L} \sum_{f \in F} \beta_{jk} z_k^{ft} M^{mf} \lambda_{jk}^{6,lm} \leq \xi^t \quad \forall t \in T
\end{aligned} \tag{4.43}$$

It should be noted that, with the increase in the number of periods, the disaggregation of cuts actually results in an increase in the number of constraints added to the MP, resulting in the increased size of the MP. However, the disaggregation helps in exploring the solution space in a better fashion, and ultimately reduces the computational complexity and CPU time due to fewer number of iterations.

#### 4.5.2.4 Relaxed Benders Cut

In the early stage of BD algorithm, multiple cuts can be added into the MP to quickly improve the lower bound. To generate such multiple cuts, 'relaxed Benders cut' is a strategy reported in the literature (McDaniel and Devine, 1977). Following this strategy, the relaxed version of the MP is solved (that is, by relaxing the integrality requirement on the related variables) instead of the original MP. The solution of the relaxed MP is fed to the DSP, and optimality or feasibility cuts are generated. At any iteration of the BD algorithm, the relaxed Benders cut strategy would result in multiple additional cuts that are to be fed back to the original MP, and the process of the BD algorithm continues. The relaxed Benders cut is as follows:

$$\begin{aligned}
& \sum_{i \in I} \sum_{l \in L} (1 + \kappa_i^l) W_i^l \lambda_i^{2,lt} + \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} Q^{lf} z_j^{ft} \lambda_j^{4,lt} \\
& + \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} \sum_{f \in F} \alpha_{ij} z_j^{ft} M^{lf} \lambda_{ij}^{5,lt} + \sum_{i \in I} \sum_{j,k \in J} \sum_{l,m \in L} \sum_{f \in F} \beta_{jk} z_k^{ft} M^{mf} \lambda_{jk}^{6,lm} \leq \xi^t \quad \forall t \in T
\end{aligned} \tag{4.44}$$

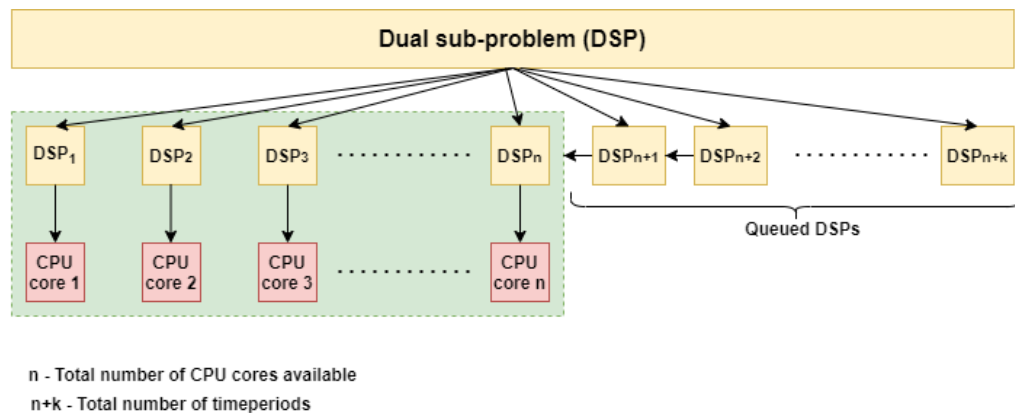
In inequality (4.44)  $\lambda_i^{2,lt}$ ,  $\lambda_j^{3,lt}$ ,  $\lambda_j^{4,lt}$ ,  $\lambda_{ij}^{5,lt}$ , and  $\lambda_{jk}^{6,lt}$  represent the dual variables obtained during relaxation of BD algorithm. Figure 4.4 depicts the framework for relaxed BD algorithm. The total number of relaxed Benders cuts generated is equal to the number of relaxed BD algorithm iterations ( $N_{relax}$ ). These cuts were also disaggregated in terms of time period  $t \in T$ . So the total number of cuts will be  $N_{relax} \times T$ . These cuts, when inserted into the first iteration of regular BD algorithm, drastically reduce the solution space and provide a better initial LB. Relaxed BD algorithm performs iterations much faster than regular BD algorithm. However, the number of iterations increases, and so the CPU time. Therefore, a suitable trade-off between the required number of relaxed Benders cuts and the amount of CPU time is sought by specifying optimality gap practical and workable.

#### **4.4.1.2.5 Parallelism**

Although the DSP is an LP problem and can be solved efficiently for small size instances, its solution time dramatically increases for medium and large size problems. Parallel computing is an effective and efficient technique to solve the DSP having a special structure (Linderoth, 2003). The DSP can be disaggregated in terms of time periods, and it can be solved individually and simultaneously for each time period. Therefore, parallel computing is used to solve the DSP simultaneously in each iteration of the BD algorithm.

To perform parallel computing, the DSP is first decomposed into multiple DSPs that are independent of the time period. Each CPU core solves one DSP (independent of the time period) and provides the value of the objective function and the dual variables. The final objective function value of the original DSP is obtained by adding time-independent objective function values of each DSP, and the values of dual variables are saved for a later iteration. If

the number of DSPs is more than the number of CPU cores available, then remaining problems will be addressed sequentially. For example, if a 4-core processor is used to solve a 10-time period problem, then there will be we 10 DSPs to solve. The first four problems are assigned to each core, while the remaining six problems are queued. As soon as a core becomes available, it will select the next problem in the queue. Figure 4.3 illustrates the framework for implementing parallel computing.



**Figure 4.3: Framework for implementation of parallelism to solve DSP**

The above-mentioned strategies are employed to accelerate the BD algorithm. Figure 4.4 depicts the proposed framework. The steps and criteria for termination are similar to those used in classical BD algorithm. The varied colours represent the proposed acceleration strategies, and the arrows represent their proper position.



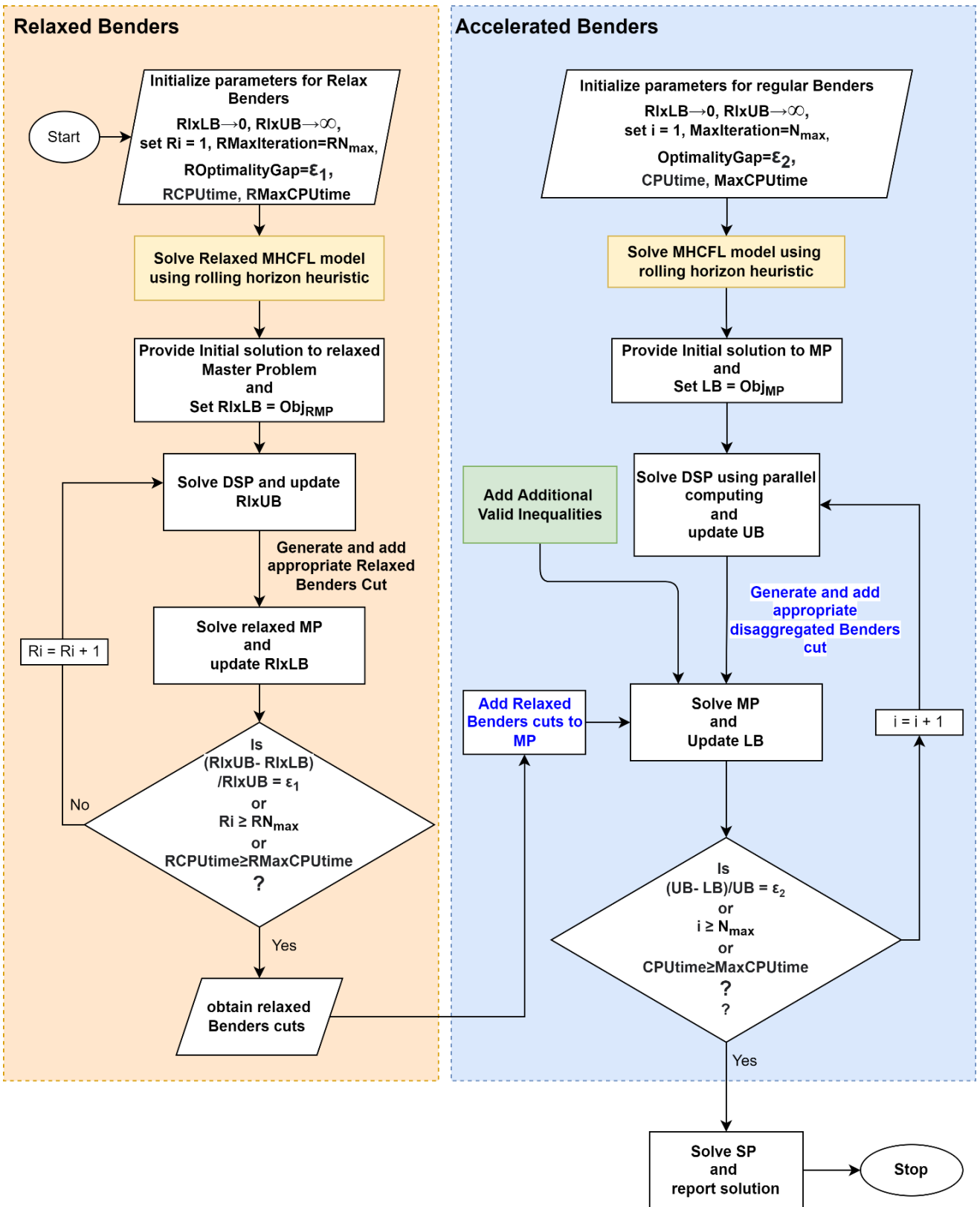


Figure 4.4: Framework for accelerated Benders decomposition algorithm

### 4.5.3 Benders type heuristic

In the implementation of the classical as well as accelerated BD algorithm, it is observed that the time required to solve the master problem (MP) increases exponentially due to its combinatorial nature. It is very time consuming to solve the MP to optimality in each iteration of the BD algorithm using the off-the-shelf solvers, and it results in slower convergence. In such cases, the MP can be solved efficiently using a heuristic. The resulting solution might not be optimal. However, with an appropriate selection of the heuristic, a good bound is guaranteed. This strategy has been named as Benders type heuristic in the literature (Santoso et al., 2005). In this work, we use a matheuristic, namely relax-and-fix (R&F), to quickly solve the MP that results in the Benders type heuristic. The relax-and-fix (R&F) is a heuristic that decomposes a large Integer Programming (IP) problem into several smaller and manageable sub-problems. As the problems are partially relaxed, IPs have only a small number of binary variables to be optimized. They are easy to solve using exact methods like a branch and bound. In the R&F heuristic, the key is managing the set of binary variables. Here, most of the binary variables are either relaxed (i.e., treated as continuous variables) or kept fixed (to their values obtained in the previous step), and only a few binary variables are kept free (i.e., to be optimized). These problems are solved sequentially to build a complete solution to the problem. It is observed that, due to their origin in solving lot-sizing problems (Pochet & Wolsey, 2006), these heuristics have been widely used in the lot-sizing, scheduling and production planning context for the last decade. The solution provided by the R&F heuristic may not be optimal, but it will provide a good quality solution in comparatively less CPU time. The steps involved in Relax and Fix heuristic are shown in Table 4.1, and the pseudo-code is

shown in Figure 4.5. In each step, the problem is to optimize till the terminating condition – optimality gap or CPU time is reached. The proposed approach works as follows.

To solve the MP using R&F heuristic, the variables related to the location of facilities are focused first. The corresponding  $y_j^{ft}$  variables are taken as binary, while this requirement is relaxed for  $z_j^{ft}$  and  $\bar{y}_j^{fst}$  variables. With a relaxed condition, the proposed MP is solved to obtain the location of facilities. In the next step, the MP is solved by setting  $z_j^{ft}$  variables as binary and keeping the values of  $y_j^{ft}$  variables obtained in Step 1 (Figure 4.5) as fixed. The solution thus obtained is ready with a set of operating and located facilities. In Step 3,  $\bar{y}_j^{fst}$  variables are kept as binary, and the MP is solved by keeping the values of other sets of binary variables fixed. This step completes the construction of the feasible solution to the MP. The numerical experiments revealed that the R&F heuristic produces near-optimal or a very good quality solution to the MP in several instances.

**Table 4.1: Steps for relax-and-fix**

Step	Binary variables related to facility		
	Location ( $y_j^{ft}$ )	Operation ( $z_j^{ft}$ )	Upgradation ( $\bar{y}_j^{fst}$ )
1	<b>Binary</b>	Continuous	Continuous
2	Fixed	<b>Binary</b>	Continuous
3	Fixed	Fixed	<b>Binary</b>

---

**Benders-type heuristic – to solve MP in each iteration of BDA**

- 1: Initialize  $(x_{ij}^l, x_{jk}^m, y_j^{ft}, z_j^{ft}, y_j^{fst} \geq 0; \overline{Obj} = 0)$
  - Step 1**
  - 2: Set  $y_j^{ft} \in \{0,1\}, \forall j \in J, t \in T$
  - 3: Solve MP by the solver
  - 4: Set  $\overline{Obj} \leftarrow Obj_{MP}$  and  $y_j^{ft} \leftarrow y_j^{ft}, \forall j \in J, t \in T$
  - Step 2**
  - 5: Set  $z_j^{ft} \in \{0,1\}, \forall j \in J, t \in T$
  - 6: Add a constraint  $y_j^{ft} = y_j^{ft}, \forall j \in J$  to MP
  - 7: Solve MP by the solver
  - 8: Update  $\overline{Obj} \leftarrow Obj_{MP}$  and  $\tilde{z}_j^{ft} \leftarrow z_j^{ft}, \forall j \in J, t \in T$
  - Step 3**
  - 9: Set  $y_j^{fst} \in \{0,1\}, \forall j \in J, f \in F, g \in F, t \in T$
  - 10: Add a constraint  $z_j^{ft} = \tilde{z}_j^{ft}, \forall j \in J, f \in F, t \in T$  to model MP
  - 11: Solve MP by the solver
  - 12: Update  $\overline{Obj} \leftarrow Obj_{MP}$  and  $\tilde{z}_j^{ft} \leftarrow z_j^{ft}, \forall j \in J, t \in T$
  - 13: Remove constraints added in Steps 6 and 10 from the MP
- 

**Figure 4.5: Pseudo-code for proposed Benders type heuristic**

## 4.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is one of the popular evolutionary population-based stochastic optimization techniques developed by Kennedy and Eberhart (1997). PSO is inspired by the social behaviour of birds flocking in their search for food and shelter. In PSO, a swarm consists of a number of interactive particles that fly around the high-dimensional search space. Each particle represents a potential solution to the problem. Each particle has a randomly initialized position and velocity. The position and velocity of the particle are updated using the particle's own experience and also swarm experience to allow it to fly to the next position. The iterative procedure of updating the position and velocity of the particle improves the quality of the solution of each particle calculated based on a fitness function. PSO has been successfully applied for various optimization problems. Kennedy et al. (1997) have also

developed a binary version of PSO for discrete optimization problems. In binary PSO, the position of a particle does not completely depend on the velocity but also on the random number. Binary PSO has also been used by many other researchers for various combinatorial optimization problems. Taşgetiren and Liang (2004) applied the binary PSO for the lot-sizing problem. A modified multi-objective PSO for healthcare facility planning was implemented by Elkady and Abdelsalam (2016). Kaushik et al. (2020) applied binary PSO to improve the performance of wireless sensor networks. Various other researchers (Ali et al., 2020; Guo et al., 2014; Premalatha and Natarajan, 2008; Zhao et al., 2010) have also worked to improve the performance of PSO using local search strategy for discrete problems. After going through the literature review carried out by Basu et al. (2015), it is observed that the number of applications of binary PSO on discrete facility location problems is still small. In this chapter, binary PSO with local search is applied to solving the multi-period facility location-allocation problem for determining the location of the facilities. Binary PSO is hybridized with the Gurobi solver for obtaining allocation decisions. The local search helps in improving the quality of solutions obtained from binary PSO.

The mathematical model presented in Section 4.3 can be decomposed into two sub-problems: (i) location and (ii) allocation problems. The location problem will consist of three binary variables:  $y_j^{ft}$ ,  $\bar{y}_j^{fgt}$  and  $z_j^{ft}$ .  $z_j^{ft}$  derives its values from the other two binary variables. Therefore, binary PSO deals with only  $y_j^{ft}$  and  $\bar{y}_j^{fgt}$ . The value of  $z_j^{ft}$  will be decided based on the values of  $y_j^{ft}$  and  $\bar{y}_j^{fgt}$ . Thus, the problem of the location is to be solved by binary PSO and will determine the values of  $y_j^{ft}$  and  $\bar{y}_j^{fgt}$ .  $z_j^{ft}$  will be assigned a value from the values of these two variables. The allocation problem will involve variables as  $x_{ij}^{lt}$ ,  $x_{jk}^{lmt}$ ,  $\bar{x}_j^{lt}$  and  $z_j^{ft}$ .

In the initial phase of the development of binary PSO, it was observed that the random allocation to the facilities resulted in a small value of the fitness function. A large number of iterations were required to improve the fitness function value for a very large size of population. A large number of iterations required very high computational time. So, the random allocation of MTBs was not computationally efficient and thus not suitable. Allocation of MTBs is thus carried out using Gurobi solver. It is observed that it takes very little time to provide a better solution. The framework of binary PSO, proposed as a solution strategy for the MHCFL model, is shown in Figure 4.6 and its various steps are detailed below.

### ***1. Parameter initialization***

The values for the parameters of binary PSO were decided after conducting various computational experiments for widely random values assigned to parameters of a medium-size problem and are taken as,

Swarm size (Number of particles) = 10,

Maximum number of iterations = 100,

Inertia weight parameter ( $C_1$ ) = 0.1,

Cognitive parameter ( $C_2$ ) = 0.1, and

Social parameter ( $C_3$ ) = 0.1.

Each particle ( $p$ ) has three parameters, namely, position ( $(y_{jp}^{ft}, \bar{y}_j^{fst})$ ), velocity ( $(v_p)$ ), and the particle's best position ( $p_{best}$ ). The overall best position of the particles having the best fitness value is called the global best ( $g_{best}$ ) position (Yang et al., 2014). The position of each particle

in the search space is represented by a set of coordinates. During the search, the current positions of all the particles are evaluated using the fitness function.

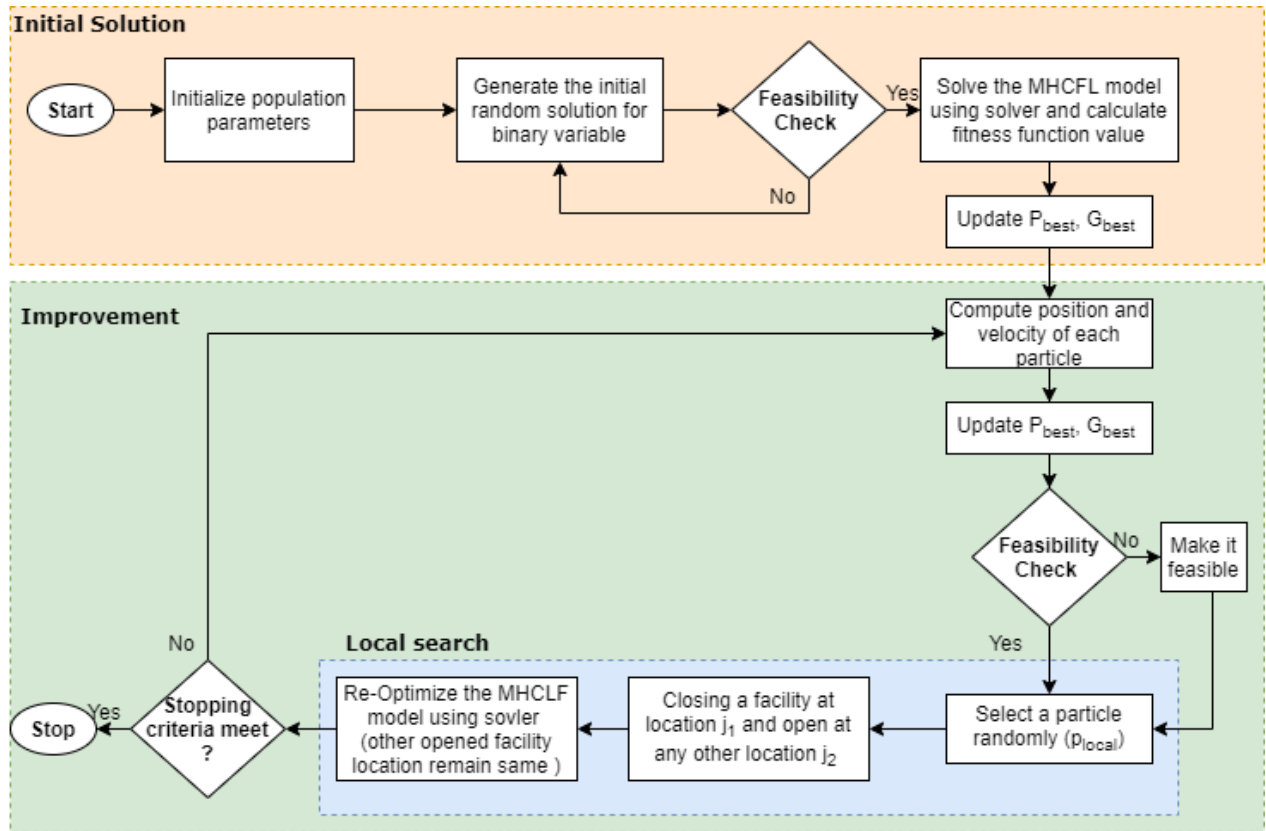


Figure 4.6: Flowchart showing binary PSO implementation

## 2. Generation of random initial solution

The initial solution is generated using a greedy heuristic, detailed in Figure 4.7. First, the total demand and the total availability in the first period of the planning horizon are determined for each service type. Suppose the demand is more than the availability of high-level services. In that case, the lower-level facilities are first upgraded to higher-level facilities until the available capacity for the higher-level service type is at least equal to the corresponding demand. Upgradation is first worked out as it is economical compared to the opening of new facilities. Even after exhausting the option of up-gradation of facilities, it may be found that the demand

is still more than the availability. If so, the option of addition of facilities is exercised. Upgradation is carried out randomly, but first looking into the demand of service type 3 and then for service type 2. To meet the demand for service type 3, the up-gradation of SC to CHC is carried out first. After all the SCs have been tried, then PHCs are taken up for up-gradation. After resolving the issue with the demand for service type 3, service type 2 is taken up in a similar manner. In case demand is in excess of the availability even when the up-gradation option is completely exhausted, the new facilities are established at new locations. While locating new facilities, first, the demand of service type 3 is focused that has to be satisfied by establishing CHCs, thereafter for service type 2 by PHCs and SCs in the last.

For the subsequent time periods, the determination of required facilities of each type is calculated based on the capacity of existing facilities in the previous time period and the demand during the period. If the existing facilities are not sufficient to meet the demand, then a decision regarding up-gradation of existing facilities or opening of new facilities will be taken up as done earlier. Following this strategy, each particle will represent a random initial solution  $(y_{jp}^{ft}, \bar{y}_j^{fgt})$  that will be feasible and good.

### ***3.Calculation of fitness function***



The current position ( $y_{jp}^{ft}$ ,  $\bar{y}_j^{fgt}$  and  $z_{jp}^{ft}$ ) of each particle and the value of allocation variables ( $x_{ij}^{lt}$ ,  $x_{jk}^{lmt}$ ,  $\bar{x}_j^{lt}$ ) are used to calculate the fitness value. The fitness value of each particle in each iteration is evaluated using expression (4.1) of the MHCFL model.

#### **4. Updation of particle best and global best**

The fitness value of each particle, determined in the previous step, is then compared with the value corresponding to the current position and the best position of the particle. If the current position of the particle has a better fitness function value compared with the previous best, then the best position of the particle, i.e.,  $p_{best}$  is updated with this value. The same scheme is followed for all the particles. The global best ( $g_{best}$ ) position of the swarm is also updated if found better than the previous one.

#### **5. Calculating particle velocity and position**

The velocity of a particle ( $v_p$ ) in the next iteration ( $k+1$ ) is determined using Equation (4.45), where  $c_1$  is the inertia weight,  $c_2$  the cognition learning factor,  $c_3$  the social learning factor, and  $r_1$  and  $r_2$  are the uniformly generated random numbers in the range of [0, 1]. Further, each particle will move to the next position ( $y_{jp}^{fk+1}$ ,  $\bar{y}_{jp}^{fgtk+1}$ ) by adding the velocity ( $v_p^{t+1}$ ) to the current position ( $y_{jp}^{fjk}$ ,  $\bar{y}_{jp}^{fgtk}$ ) following Equation (4.46).

$$v_p^{k+1} = c_1 v_p^k + c_2 r_1 (p_{best}_p - y_p^k) + c_3 r_2 (g_{best} - y_p^k) \quad (4.45)$$

$$y_p^{t+1} = y_p^t + v_p^{t+1} \quad (4.46)$$

After adding the velocity to the particle, it is observed that the binary variables may assume non-binary values. To deal with this issue, both piece-wise linear function and sigmoid

function are used to update the position of the particle. These functions are detailed in respective Equations (4.47) and (4.48), respectively. Next, the position of the particle is updated using Equation (4.49) that ensures the position of a particle to be 0 or 1. The velocity and position calculation discussed above is also followed for the up-gradation variable ( $\bar{y}_{jp}^{fgt}$ ).

$$y_{jp}^{fik} = \begin{cases} 1, & \text{if } y_{jp}^{fik} \geq 1 \\ y_{jp}^{fik}, & \text{if } 0 < y_{jp}^{fik} < 1 \\ 0, & \text{if } y_{jp}^{fik} \leq 0 \end{cases} \quad (4.47)$$

$$\text{sigmoid}(y_{jp}^{fik}) = \frac{1}{1 + e^{-y_{jp}^{fik}}} \quad (4.48)$$

$$y_{jp}^{fik} = \begin{cases} 1, & \text{if } U(0,1) < \text{sigmoid}(y_{jp}^{fik}) \\ 0, & \text{Otherwise} \end{cases} \quad (4.49)$$

## 6. Feasibility check

A feasibility check is carried out to ensure the feasibility of the solution represented by the positions of the particles obtained in the previous step. A feasibility issue will be encountered due to the restriction on the coverage distance, be it referral or otherwise. An infeasible solution means some of the locations are not covered by the existing facilities planned so far. To arrest this problem, some new facilities are to be established at locations with no facilities. The type of facility selected will depend upon the service type, demand from those locations, etc.

## 7. Local search

After ensuring a feasible solution corresponding to each particle, a local search is conducted to ameliorate the fitness value further. A local search can be performed for all the particles or taking a few ones (Premalatha and Natarajan, 2008). Since swarm size is large, performing a local search for all the particles will consume a lot of time. Therefore, the local search is carried

out with only 10% of the total particles. The information of the fittest particle will be shared with other particles to help in improving their positions and fitness values. The local search involves moving the facility being planned for the current location to be established at a new location through further optimization using the MHCFL model. Following the local search iteratively, we try to improve the fitness value of the particles. The pseudo-code for the local search is shown in 4.8.

### ***8.Stopping criterion***

The algorithm is terminated if the limit on either the maximum number of iterations or the maximum time limit is reached, whichever is earlier. The best-known fitness function value ( $g_{best}$ ) and CPU time are reported.

**Figure 4.7: Steps for generation of random initial solution**

---

Step 0:	<b><i>For time period <math>t = 1</math>:</i></b>
Step 1:	Calculate the total demand and availability for each service type
Step 2:	Upgrade existing facilities randomly
Step 3:	Determine the number of CHCs required for service type 3 (based on the demand of service type 3 and availability with existing CHCs)
Step 4:	Determine the number of PHCs required for service type 2 (based on the demand for service type 2 and availability with existing PHCs and CHCs, including that added in step 3)
Step 5:	Determine the number of SCs required for service type 1 (based on the demand for service type 1 and availability with existing SCs, PHCs and CHCs and including that added in steps 3 and 4)
Step 6:	Locate the CHCs, PHCs, and SCs randomly identified in steps 3, 4 and 5
Step 7:	<b><i>if randomly-opened facilities cover all the locations:</i></b>
Step 8:	Allocate the MTBs to the facilities and move to the next step <b><i>else:</i></b> Open new required facilities and repeat step 7
Step 9:	<b><i>If all the locations are covered (for both referral and non-referrals):</i></b> Move to the next step <b><i>else:</i></b> Open new required facilities and repeat
Step 10:	<b>Set <math>t = t + 1</math></b>
Step 11:	<b><i>For time period <math>t</math> :</i></b>
Step 12:	<b><i>if the availability with the existing facilities is sufficient to fulfil the demand :</i></b>
Step 13:	Allocate the MTBs to the required facility type <b><i>else:</i></b>
Step 14:	Upgrade existing facilities or open new required facility-type randomly

Step 15: Repeat step 10 onwards until the whole planning horizon is covered

---

**Figure 4.8: Pseudo-code for local search**

---

<b>Pseudo-code for local search</b>	
<b>Step 0:</b>	Randomly select particle ( $p_{local}$ ) for local search
<b>Step 1:</b>	Initialize $\overline{y}_j^{ft} = 0, \forall j \in J_n, \forall f \in F, \forall t \in T$
<b>Step 2:</b>	set $\overline{y}_j^{ft} = y_{jP_{local}}^{ftk}, \forall j \in J_n, \forall f \in F, \forall t \in T$
<b>Step 3:</b>	select randomly $\overline{y}_{j_r}^{ft}$ , if $\overline{y}_j^{ft} = 1$
<b>Step 4:</b>	add constraint $y_{j_r}^{ft} = 0, \forall j_r \in J_n, \forall f \in F, \forall t \in T$ if $\overline{y}_j^{ft} = 1$ :
<b>Step 5:</b>	add constraint $y_j^{ft} = \overline{y}_j^{ft}, \forall j \in J, \forall f \in F, \forall t \in T$ add constraint $\overline{y}_{j_r}^{fg\tau}$ and $z_{j_r}^{f\tau} = 0, \forall \tau \geq t \in T$
<b>Step 6:</b>	Set timelimit = $t_0$
<b>Step 7:</b>	Optimize MHCFL model
<b>Step 8:</b>	<b>Report</b> (Fitness value) $_{P_{local}}$ $\leftarrow$ Objective value and $y_{jP_{local}}^{ftk} \leftarrow y_j^{ft}$

---

## 4.7 Hybridized Simulated Annealing

Simulated Annealing (SA) is a metaheuristic developed by Kirkpatrick et al. (1983). SA was inspired by an analogy between the physical annealing process of metals and the problem of solving large combinatorial optimization problems. Annealing requires metals to be cooled from being at a high temperature to a desired temperature following a cooling schedule. SA, following this approach, starts with some solution and, seeks improvement in the crystalline structure of the metal (solution quality) following a cooling schedule (going through some iterations). The process is terminated when desired low temperature (or the desired quality of the solution) is obtained. In the recent decades, SA has been used for solving numerous combinatorial optimization problems such as crew scheduling and rostering problems (Emden-Weinert and Proksch, 1999; Hadiani et al., 2014), vehicle routing problems (Czech and

Czarnas, 2003; Wang et al., 2015; Zhou et al., 2018), facilities location problem (Arostegui et al., 2006), Knapsack problem (Dantas and Cáceres, 2018), facility layout problem (Defersha and Hodiya, 2017) and many more. Ghaderi and Jabalameli (2013) developed a simulated annealing approach to solve healthcare facility network design problems using in conjunction Fix-and-Optimize heuristic. Recently, Turhan and Bilgen (2020) proposed a hybrid approach (using Fix-and-Relax and Fix-and-Optimize into Simulated Annealing) to solve a nurse rostering problem. They generated the initial solution using the Fix-and-Relax approach, to be used by SA. Further, 2-exchange, 3-exchange, multi-exchange, shift-on, and shift-off concepts were applied to improve the solution quality. The Fix-and-Optimize method was applied after no improvement was witnessed. Looking into the benefits of hybridisation, the present work uses the fix-and-optimise approach along with SA. The proposed hybrid approach has not been applied so far to solving the maternal healthcare problem. Since SA was reported to perform better if it starts with a good initial solution, fix-and-optimise (F&O) heuristic was used to feed a good quality solution to the proposed SA. However, the problem is first solved as a set covering problem before applying F&O heuristic to reduce the search space. This improves the quality of the solution to be provided by the F&O heuristic. The proposed hybrid FO & SA approach is discussed in further sub-sections.

#### **4.7.1 Set covering problem**

The initial solution of the relaxed MHCFL problem is determined by treating it as a set covering problem. In Section 4.3, the restrictions in terms of maximum coverage distance ( $d_l$ ) are incorporated in the MHCFL model using a binary parameter ( $\alpha_{ij}$ ). In order to cover all the MTBs, at least one facility should operate within the maximum coverage distance of each location of MTBs. From this perspective, a set covering model is applied to the location

problem to determine the minimum number of locations of the healthcare facilities. The binary variables,  $\bar{z}_j$ , are taken to formulate the set covering problem. The other notations used in the set covering formulation have the same definition as described in Section 4.3.

$$\bar{z}_j = \begin{cases} 1, & \text{if a healthcare facility is located in } j^{\text{th}} \text{ location,} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{Minimize } \sum_{j \in J} \bar{z}_j \quad (4.50)$$

$$\sum_{j \in J} \alpha_{ij} \bar{z}_j \geq 1 \quad \forall i \in I \quad (4.51)$$

$$\bar{z}_j = 1, \quad \forall j_e \in J \quad (4.52)$$

The objective function (4.50) minimizes the total number of locations for healthcare facilities to cover all the locations. Constraint (4.51) stipulates that each location ( $i$ ) must be covered by at least one facility. Constraint (4.52) shows that the facility will remain operating in future if it is already existing.

#### 4.7.2 Fix-and-optimize

The set-covering problem provides locations of the healthcare facilities without considering allocation decisions (step '0' of Figure 4.9). To start with, it is assumed that each of these selected locations will have CHCs in the first period of the planning horizon (Steps 1 to 17 of Figure 4.9). While solving this problem with this assumption, the selected locations with facility type III (CHC) will be capable (because of availability of all service types) of meeting the demand from the nodes covered within the maximum permissible distance. It may so happen that the current solution may have a very high cost due to high travel, penalty, establishment, and operating costs. To reduce this cost, the current solution may require lower-

level facilities (PHC or SC) replacing a CHC planned to be at a location, or altogether a new location may be chosen instead of establishing SC, PHC or CHC, as the case may be. It will be illogical to consider the existing facilities for possible removal. Such facilities will be considered only for upgradation at some later period of the planning horizon. For solving the associated problem, the MHCFL formulation is not used as such because of the resulting large size of the formulation. Instead, all the locations are chosen one by one, considered the same as a potential location for a healthcare facility while freezing the earlier location decisions. It is carried out irrespective of whether a nonexistent facility (which will be a CHC) was located in it earlier. With this consideration incorporated into MHCFL formulation, now mainly the allocation problem has to be solved while identifying the suitability of the location under consideration to have the right type of facility with possible up-gradation at a later period. It is reiterated that when a facility already existed at the location under consideration, the problem is solved while acknowledging the facility-type existing there. This process is iterated until no improvement in objective function value is witnessed going through all the locations. Because of the combinatorial and complex nature of the problem at this stage, the termination of this iterative procedure may not be witnessed in a reasonable amount of time. In view of this, it is logical to set a limit on the total CPU time for this stage. For the experimental analysis detailed in Section 4.5, a time limit of 2 hours was considered.

**Figure 4.9: Pseudo-code for fix-and-optimize**

Steps	Description
Step 0:	<i>Use the solutions obtained from solving set covering problem for initializing the variables</i>

---

Step 1: *Initialize local parameters*  $zstatus_j^{ft} = 0, ystatus_j^{ft} = 0$  and  $\overline{ystatus}_j^{fgt} = 0 \quad \forall j, f, g, t$

Step 2: *while*  $j \in J$  *do*

Step 3:     *if*  $\overline{z}_j == 1$ :

Step 4:         *if*  $f == 3$ :

Step 5:              $zstatus_j^{ft} = 1 \quad \forall t \in T$

Step 6: *while*  $j \in J_n$  *do*

Step 7:     *if*  $\overline{z}_j == 1$ :

Step 8:          $f == 3$ :

Step 9:             *if*  $t == 1$ :

Step 10:                  $ystatus_j^{ft} = 1$

Step 11: *while*  $j \in J_e$  *do*

Step 12:     *if*  $\overline{z}_j == 1$ :

Step 13:         *if*  $t == 1$ :

Step 14:             *if*  $f \leq 2$ :

Step 15:                 *if*  $k_j^{ft} == 1$ :

Step 16:                     *if*  $g == 3$ :

Step 17:                          $\overline{ystatus}_j^{fgt} = 1$

Step 18: *Taking the above input, Now F & O approach is started*  
*as Iteration Counter (IC) = 0 and Obj<sub>min</sub>*

Step 19:     *for*  $j \in J$ :

Step 20:          $y_l^{ft} = ystatus_l^{ft} \quad \forall l \in J, f \in F, t \in T$  *where*  $l \neq j$

Step 21:          $z_l^{ft} = zstatus_l^{ft} \quad \forall l \in J, f \in F, t \in T$  *where*  $l \neq j$

Step 22:          $\overline{y}_l^{fgt} = \overline{ystatus}_l^{fgt} \quad \forall l \in J, f \in F, g \in F, t \in T$  *where*  $l \neq j$

Step 23:     *Solve MHCFL model and obtain the value of*  $z_j^{ft}, y_j^{ft}$  *and*  $\overline{y}_j^{fgt} \quad \forall j, f, g, t$

Step 24:     *update*  $zstatus_j^{ft} \leftarrow z_j^{ft}, ystatus_j^{ft} \leftarrow y_j^{ft}$  *and*  $\overline{ystatus}_j^{fgt} \leftarrow \overline{y}_j^{fgt} \quad \forall j, f, g, t$

Step 25:     *Return*  $zstatus_j^{ft}, ystatus_j^{ft}$  *and*  $\overline{ystatus}_j^{fgt} \quad \forall j, f, g, t$

---

### 4.7.3 Implementation of Simulated annealing

Because of (i) the sequential approach used in the F&O method and (ii) the time limit on the CPU time, the F&O stage may not necessarily yield an optimal or a good quality solution. In order to further improve the solutions, the simulated annealing (SA) method is applied,



proceeding with the current solution obtained from the F&O to generate the neighbourhood solutions in the search for an improved solution. The flowchart showing the framework of the proposed hybridized simulated annealing is provided in Figure 4.10. Various important strategies and components of the proposed SA are detailed below.

### **Neighbourhood search**

At this stage, three sets of locations will be formed. The first set (set I) will be of the locations having the existing facilities, the second one (set II) will have those locations that are going to house new facilities, and the third one will be of locations without any allocated facilities. The process of identifying neighbourhood locations is also shown in Figure 4.11. The proposed SA follows a cooling schedule where control on the cooling is carried out at some locations but choosing them randomly. For this purpose, a node from set II (planned currently to house the facilities) is chosen randomly. Next, all the unassigned nodes within their coverage distance are identified. Out of all these nodes, only those nodes are considered for swapping the facilities from the node under consideration that will not violate the restrictions on the maximum travel distance either by incoming or outgoing referrals. Now, keeping all the earlier decisions intact, removing the current node from consideration, and taking all identified neighbourhood locations as candidates for possible swapping, the problem is solved using the MHCFL formulation. Naturally, the improved solution is retained. However, an inferior solution is also accepted based on a probability value calculated using the energy magnitude equation with the Boltzman constant taken as 1 (Liang, 2020). The current cooling schedule completes in two parts, each of 10 iterations. Acceptance of an inferior solution is sometimes carried out to escape from the local minima and to explore the search space. The acceptance of an inferior solution requires a uniformly distributed random number  $r \sim U[0, 1]$  with value

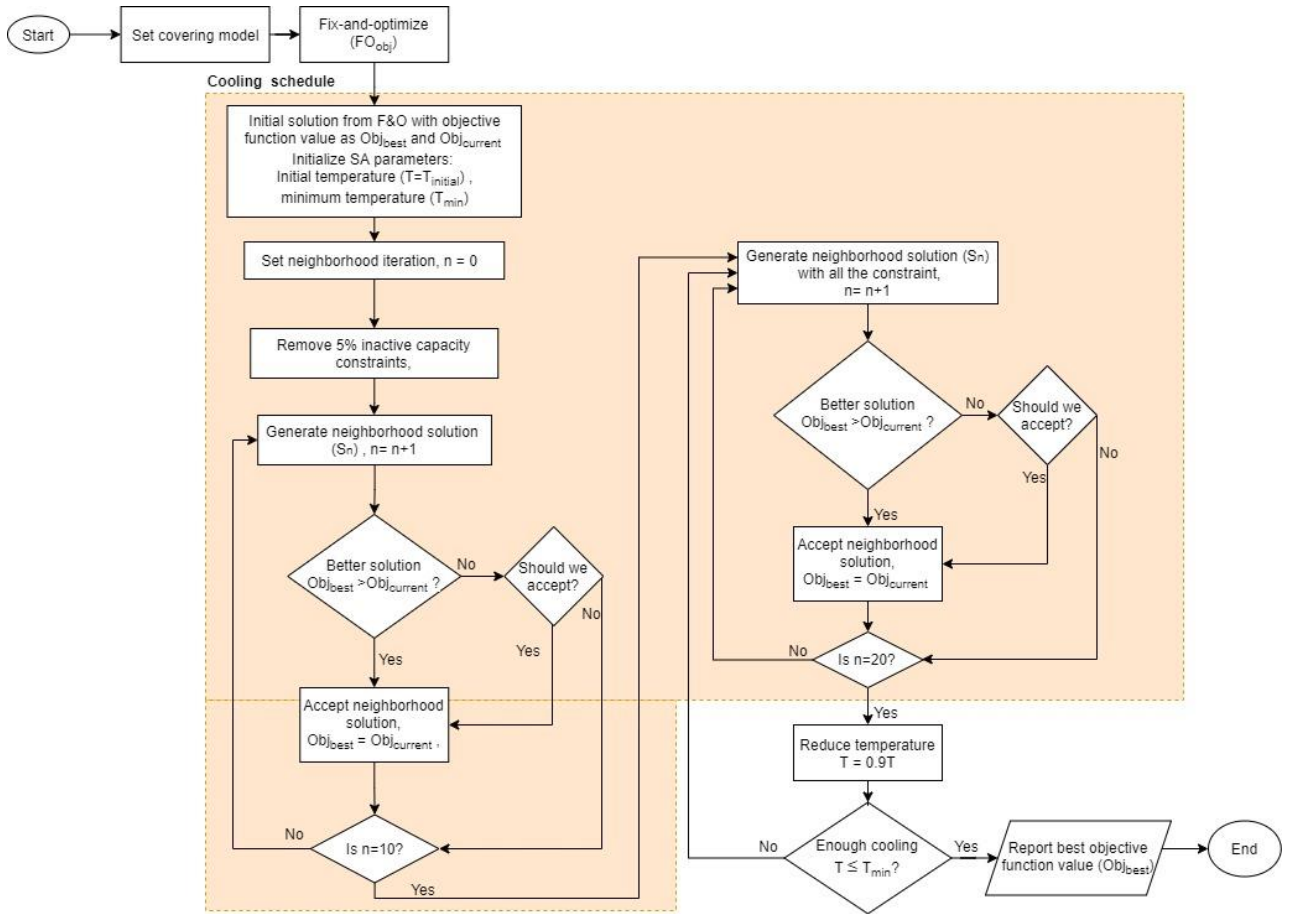
' $r'$ ' value to be more than the probability value defined as  $P = e^{\frac{-\delta E}{KT}}$  with  $K$  equal to 1. The probability depends on the current temperature ( $T$ ) and the amount of degradation ( $\delta E$ ) taken as the difference between objective function values of the best solution obtained so far and the generated neighbourhood solution. During the initial iterations, when the temperature is high, the probability of accepting the worst solution is high. As the SA algorithm progresses, the temperature comes down, and thus, the probability of accepting the inferior solution decreases. At a particular temperature, 20 iterations are performed for computational experimentation, with the probability of accepting the solution being high if the amount of degradation is low.

### **Use of removal of inactive capacity constraints**

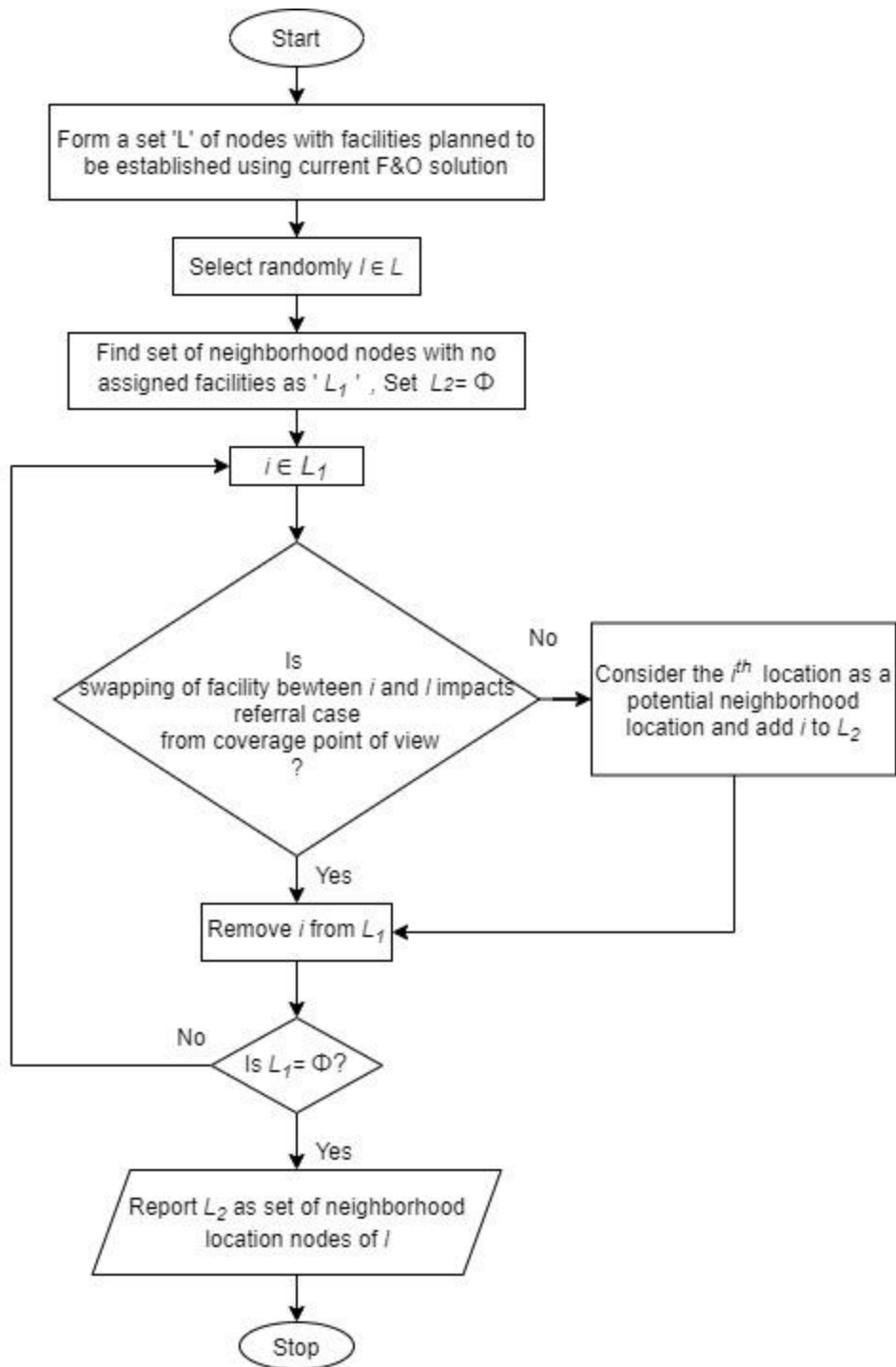
The annealing schedule is followed in two steps. The first half iterations go with some inactive capacity constraints left out. While in the remaining half, all the capacity constraints are considered. The capacity constraints that are neglected in the first half are a maximum of 5% of the total number of such inactive constraints having higher slack than those that are retained in the current solution. While working on a number of problems, such a strategy was found to result in a good solution in a comparatively lesser amount of CPU time.

### **Defining initial and final temperatures**

The initial temperature is kept very high and is fixed at a value while considering a worst-case scenario of having too many facilities to completely meet the demand. It is assumed that MTBs do not violate the restriction on the maximum coverage distance. While working with various hypothetical problems with a varying number of nodes, setting the initial temperature using the method described above and reducing it by 10% until attaining a low temperature of 100, the SA approach was found to provide a good result in general.



**Figure 4.10: Flowchart of proposed simulated annealing framework**



**Figure 4.11: Framework for the generation of neighbourhood solutions**

## 4.8 Computational Experiments

In this section, the performance of the proposed solution approaches has been studied. For solving the mathematical models, the Gurobi optimization solver was used with an interface in Python 3.8 (Rossum and Drake, 2011). Other solution approaches were also developed in Python 3.8 interface. Extensive computational experiments were performed on the supercomputer "PARAM-SHIVAY" with configuration as 2\* Intel Xeon SKL G-6148, 4 core, 2.4GHz with a memory of 192 GB, DDR4 2666 MHz. The generation of test instances and computational results are discussed in subsequent sub-sections.

### 4.8.1 Generation of test instances

The computational experiments were conducted with problems of varying sizes. The size of the problem is mainly governed by the length of the planning horizon and the number of locations. For the experimentation, four planning horizons having lengths of 5, 10, 15, and 20 time periods and the number of locations as 50, 100, 200, and 300 were taken. These problem classes are designated as D50, D100, D200, and D300. Thus a total of 16 different sizes of problem classes with five instances of each (total 80 problems) were generated to have meaningful insights from the computational experiments. Based on the number of binary and general variables, the problems were grouped into three categories, i.e., small, medium, and large, as shown in Table 4.2 and Figure 4.12. The spatial coordinates of the demand nodes or the locations of MTBs were generated randomly following Uniform Distribution:  $U[0, 200]$ . For the purpose of experimentation, the distance between two nodes  $i$  and  $j$  is taken as Euclidean distance. However, the proposed model and solution approaches are not limited to the consideration of Euclidean distance. The cost of visiting a facility is taken to be proportional to the distance. To be close to reality in the Indian context, maternal healthcare

facilities are assumed to exist in only 20% of the locations. The type of facilities existing at these locations is decided randomly. A location is assumed not to have more than one facility. For the first time period, the demand for service types 1, 2, and 3 at each location are taken as random numbers following Uniform Distribution as  $U[50, 1000]$ ,  $U[20, 400]$ , and  $U[1, 20]$ , respectively. The numbers taken were representative of the real situation where the requirement of service type 1 is the highest, and that of service type 3 is the lowest. For the subsequent time periods, the constant growth rate ( $\kappa_i^t$ ) of 1% in demand is considered for all the service types. The proportion of referrals from service types 1 to 2, 1 to 3, and 2 to 3 were taken as 0.1, 0.03, and 0.05, respectively. The maximum distance that an MTB can travel to reach a facility from her origin (i.e., coverage distance) was taken as 120 units, while in the case of referral as 150 units. The capacity for various service types at the three facility types, along with establishment, up-gradation, and operating costs applicable for the first time period, are shown in Table 4.3. A facility is taken to be of a typical size. Therefore, the capacity for a service type at a facility type does not grow with time. Whereas establishment, up-gradation, and operating costs are taken to increase at a rate of 4% per period because of inflation.

**Table 4.2: Various problem sizes**

Binary	Continuous	Time periods		Nodes	Problem class	Problem size
		in planning horizon				
1456	56670	5	50	1		
3045	212895	5	100	2		
7200	751500	10	50	5		Small
10800	1127250	15	50	9		
14400	1503000	20	50	13		
14520	3003000	10	100	6		
14610	6003000	5	200	3		
21780	4504500	15	100	10		Medium
21960	13504500	5	300	4		
29040	6006000	20	100	14		
29220	12006000	10	200	7		
43830	18009000	15	200	11		
43920	27009000	10	300	8		
58440	24012000	20	200	15		Large
65880	40513500	15	300	12		
87840	54018000	20	300	16		

**Table 4.3: Values of problem parameters used in computational experiments**

Parameters	Facility type		
	SC	PHC	CHC
<b>Capacity</b>			
Service type 1	1000	1200	1500
Service type 2	0	1000	1200
Service type 3	0	0	300
<b>Establishment cost</b>	1000000	5000000	10000000
<b>Upgradation cost</b>	SC	2000000	5000000
	PHC	-	3000000
<b>Operating cost</b>	25000	125000	250000

*\*each data is in units*

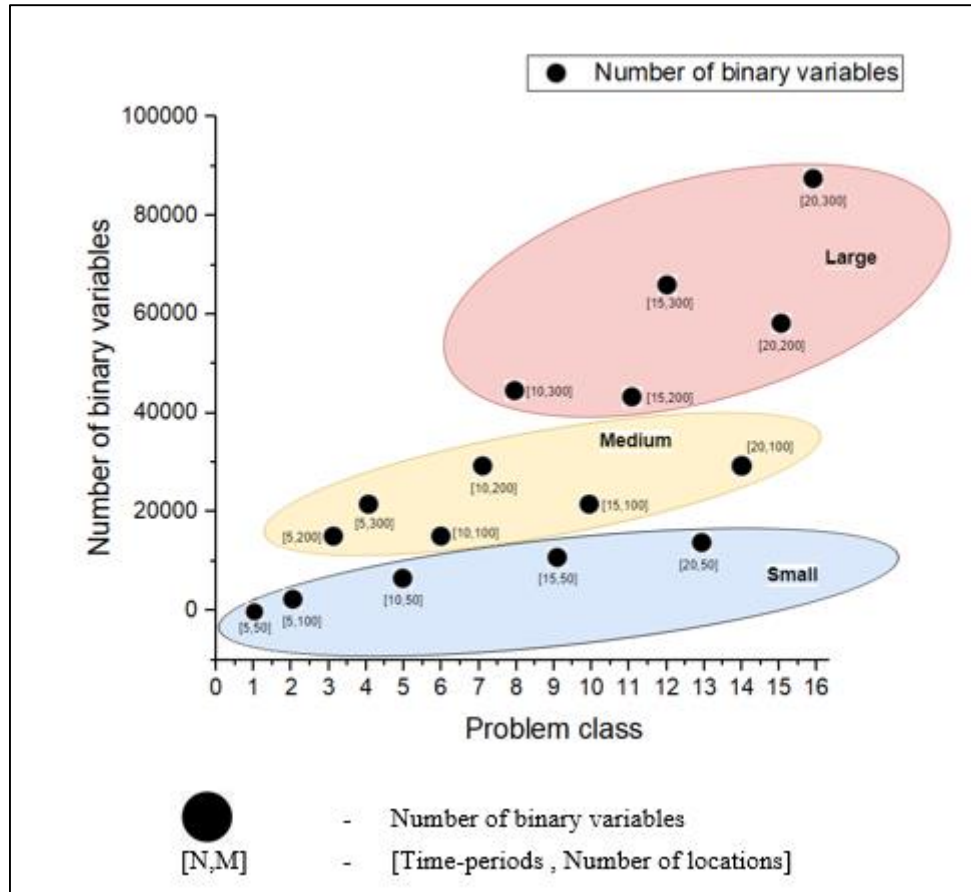


Figure 4.12: Various problem sizes

#### 4.8.2 Experimental results

The experimental results from the use of the approaches proposed in Section 4.4 are provided in Table 4.4, Table 4.5, Table 4.6, Table 4.7, Table 4.8, Table 4.9, Table 4.10 and Table 4.11 for respective planning horizon lengths of 5, 10, 15, and 20 years. A maximum CPU time limit of 24 hours is set for experimentation for each problem instance in the case of Gurobi solver, classical BD algorithm, accelerated BD algorithm and Benders type heuristic. Whereas this limit is reserved as 12 hours for hybridized SA and PSO.

In these tables, the columns indexed with number as 7, 11, 15 and 19 are labelled as 'Gap' showing the amount of gap between upper bound (UB) and lower bound (LB) values of



the objective function as a percentage of UB obtained from the MHCFL model using Gurboi, classical Benders, accelerated Benders and Benders type heuristic, respectively. The UB obtained from these approaches is then compared with the solution obtained from hybridized SA and PSO. The comparison was made based on the UB value as the problem seeks to minimize the overall cost and UB corresponds to a feasible solution. The difference in UB values obtained from the two solution approaches is reported in terms of percentage. The columns indexed as 22, 23, 24 and 25, labelled as 'D<sub>Gurobi-hSA</sub>', 'D<sub>CBD-hSA</sub>', 'D<sub>ABD-hSA</sub>', and 'D<sub>BDh-hSA</sub>', represent the percentage deviation between the objective value obtained from the application of hybridized SA and the UB values obtained from MHCFL model using Gurboi, classical ss, accelerated Benders and Benders type heuristic, respectively. The columns indexed as 28, 29, 30, 31 and 32, labelled as 'D<sub>Gurobi-PSO</sub>', 'D<sub>CBD-PSO</sub>', 'D<sub>ABD-PSO</sub>', 'D<sub>BDh-PSO</sub>', and 'D<sub>hSA-PSO</sub>' represent the percentage deviation between objective function values obtained from the application of PSO and the UB value obtained from MHCFL model using Gurboi, classical Benders, accelerated Benders, Benders type heuristic and hybridized SA, respectively. A positive value in these columns indicates that the solution obtained using approach 1 is inferior to approach 2; while a negative value suggests that approach 2 outperforms approach 1. For the large size problem instances, the comparison of the solution from the use of Gurobi was not possible due to the non-encountering of any feasible solution within the considered maximum CPU time limit. Therefore, these values are not reported in some cases.

**Table 4.4: Computational results for 5-time period**

Problem size	Nodes	Instance	Problem number	Gurobi				Classical Benders				Accelerated Benders				Benders Type Heuristic			
				LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)
small	50	1	1	19884302	19884302	0.00	238	16718270	20047847	16.61	86400	19404285	20321149	4.73	86400	19519914	20247460	3.73	86400
		2	2	21941169	21941169	0.00	1666	19337826	21961522	11.95	86400	20946634	21956157	4.82	86400	22217460	22058543	-0.72	532
		3	3	20080105	20080105	0.00	60	17582861	20227739	13.08	86400	19895820	20149619	1.28	86400	19898126	20136030	1.20	86400
		4	4	19489989	19489989	0.00	15	19472026	19489980	0.09	22769	19231081	19500196	1.40	86400	19243224	19526692	1.47	86400
		5	5	19884448	19884448	0.00	118	17323562	20036864	13.54	86400	19591041	19884448	1.50	86400	19598270	19964558	1.87	86400
small	100	1	6	39306076	39306076	0.00	32904	26791640	45174227	40.69	86400	37702683	39527820	4.84	86400	38149553	39619401	3.85	86400
		2	7	37360158	37443217	0.22	86400	27656227	40376725	31.50	86400	36501287	37543970	2.86	86400	37162890	37566324	1.09	86400
		3	8	40532999	40642900	0.27	86400	26786640	46115272	41.91	86400	39824985	40667169	2.11	86400	40430811	40739010	0.76	86400
		4	9	35191860	35263831	0.20	86400	24761436	38156056	35.10	86400	34559865	35281948	2.09	86400	35180998	35500306	0.91	86400
		5	10	36493120	36493120	0.00	4039	24250009	40221060	39.71	86400	35687547	36508418	2.30	86400	36300479	37947086	4.54	86400
medium	200	1	11	82363888	82459771	0.12	86400	40443080	159044786	74.57	86400	81092264	82695487	1.98	86400	81759597	83254642	1.83	86400
		2	12	79470115	79782380	0.39	86401	42198072	129754569	67.48	86400	77979758	80090118	2.71	86400	78463771	80136074	2.13	86400
		3	13	81314346	81436503	0.15	86400	39593284	147942792	73.24	86400	80001376	81706586	2.13	86400	80595409	82001205	1.74	86400
		4	14	76530091	76540609	0.01	86400	46140925	99394195	53.58	86400	75046269	77071708	2.70	86400	76216822	77618431	1.84	86400
		5	15	79263136	79542084	0.35	86400	44945959	117706816	61.82	86400	77722456	79765921	2.63	86400	78371991	80041334	2.13	86400
medium	300	1	16	121879774	134330930	9.27	86400	55030716	246794888	77.70	86400	95789928	140628103	46.81	86400	98797224	140599751	42.31	86400
		2	17	125204086	133204817	6.01	86400	58258272	245770071	76.30	86400	94862186	141345791	49.00	86400	99530335	141344751	42.01	86400
		3	18	126665011	146597388	13.60	86400	62093980	243221915	74.47	86400	98162637	141389355	44.04	86400	100048341	141389355	41.32	86400
		4	19	116057765	122507002	5.26	86400	54714103	241692113	77.36	86400	95919960	129398169	34.90	86400	98237766	129394466	31.72	86400
		5	20	124112918	134051913	7.41	86401	57256531	261651915	78.12	86400	100614460	140453905	39.60	86400	104164117	140453905	34.84	86400

**Table 4.5: Computational results for 5-time period**

Problem size	Nodes	Instance	Problem number	Hybridized SA						PSO						
				Obj value	$D_{\text{Gurobi-hSA}}$ (%)	$D_{\text{CBD-hSA}}$ (%)	$D_{\text{ABD-hSA}}$ (%)	$D_{\text{BDI-hSA}}$ (%)	CPU Time	Avg Obj value	$D_{\text{Gurobi-PSO}}$ (%)	$D_{\text{CBD-PSO}}$ (%)	$D_{\text{ABD-PSO}}$ (%)	$D_{\text{BDI-PSO}}$ (%)	$D_{\text{hSA-PSO}}$ (%)	CPU Time
(1)	(2)	(3)	(4)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)	(31)	(32)	(33)
small	50	1	1	19885531	0.01	-0.81	-2.14	-1.79	2160	19884302	0.00	-0.82	-2.15	-1.79	-0.01	3740
		2	2	21946916	0.03	-0.07	-0.04	-0.51	1008	21941169	0.00	-0.09	-0.07	-0.53	-0.03	5804
		3	3	20080105	0.00	-0.73	-0.34	-0.28	375	20083503	0.02	-0.71	-0.33	-0.26	0.02	3254
		4	4	19504302	0.07	0.07	0.02	-0.11	93	19489989	0.00	0.00	-0.05	-0.19	-0.07	4238
		5	5	19884448	0.00	-0.76	0.00	-0.40	132	19884448	0.00	-0.76	0.00	-0.40	0.00	4090
small	100	1	6	39398322	0.23	-12.79	-0.33	-0.56	10715	39337475	0.08	-12.92	-0.48	-0.71	-0.15	15147
		2	7	37438126	-0.01	-7.28	-0.28	-0.34	12097	37478619	0.09	-7.18	-0.17	-0.23	0.11	27366
		3	8	40726587	0.21	-11.69	0.15	-0.03	17112	40821342	0.44	-11.48	0.38	0.20	0.23	22276
		4	9	35286326	0.06	-7.52	0.01	-0.60	9576	35265423	0.00	-7.58	-0.05	-0.66	-0.06	31042
		5	10	36498028	0.01	-9.26	-0.03	-3.82	7060	36511084	0.05	-9.22	0.01	-3.78	0.04	32657
medium	200	1	11	84397022	2.35	-46.94	2.06	1.37	43200	94449671	14.54	-40.61	14.21	13.45	11.91	41384
		2	12	82203200	3.03	-36.65	2.64	2.58	43200	89282785	11.91	-31.19	11.48	11.41	8.61	40735
		3	13	86299133	5.97	-41.67	5.62	5.24	16290	94500818	16.04	-36.12	15.66	15.24	9.50	38468
		4	14	82756635	8.12	-16.74	7.38	6.62	43200	86359040	12.83	-13.11	12.05	11.26	4.35	41475
		5	15	86270952	8.46	-26.71	8.16	7.78	43200	88587587	11.37	-24.74	11.06	10.68	2.69	39930
medium	300	1	16	128260991	-4.52	-48.03	-8.79	-8.78	28838	163314157	21.58	-33.83	16.13	16.16	27.33	41594
		2	17	133417136	0.16	-45.71	-5.61	-5.61	14426	151382466	13.65	-38.40	7.10	7.10	13.47	32792
		3	18	139078462	-5.13	-42.82	-1.63	-1.63	14420	154057096	5.09	-36.66	8.96	8.96	10.77	36841
		4	19	120884659	-1.32	-49.98	-6.58	-6.58	30629	156901524	28.08	-35.08	21.25	21.26	29.79	38390
		5	20	129406612	-3.47	-50.54	-7.87	-7.87	14426	151850940	13.28	-41.96	8.11	8.11	17.34	36950

**Table 4.6: Computational results for 10-time period**

Problem size	Nodes	Instance	Problem number	Gurobi				Classical Benders				Accelerated Benders				Bender Type Heuristic			
				LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time
				(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)
small	50	1	21	35837986	35837986	0.00	7937	26068042	36386905	28.36	86400	34356996	35837986	4.13	86400	35423454	36062700	1.77	86400
		2	22	35996895	35996895	0.00	1071	30108842	36192152	16.81	86400	34582379	36063808	4.11	86400	36264370	36219899	-0.12	1055
		3	23	36778154	36778154	0.00	2778	27389003	37230372	26.43	86400	35818360	36836073	2.76	86400	36560487	36859086	0.81	86400
		4	24	34446146	34446146	0.00	75	32009281	34996745	8.54	86400	33808897	35183146	3.91	86400	33915082	34499389	1.69	86400
		5	25	35819821	35819821	0.00	1957	26506204	36861934	28.09	86400	34960492	35819821	2.40	86400	35354795	35835103	1.34	86400
medium	100	1	26	66258080	68247924	2.92	86400	43595810	82923579	47.43	86400	65437852	68507271	4.48	86400	67137291	68812613	2.43	86400
		2	27	65776818	65776818	0.00	69400	43486020	80085831	45.70	86400	63241926	66215500	4.49	86400	64961628	67857179	4.27	86400
		3	28	70110603	71927115	2.53	86400	41715371	92714935	55.01	86400	69281290	72625049	4.60	86400	71234583	76044946	6.33	86400
		4	29	64556288	64556288	0.00	50234	39695327	81225475	51.13	86400	62196039	65260119	4.70	86400	63275822	65302710	3.10	86400
		5	30	64095196	65915313	2.76	86400	41580158	83488131	50.20	86400	63205431	66232683	4.57	86400	65232424	67967907	4.02	86400
medium	200	1	31	135711033	164055181	17.28	86400	72282998	298477364	75.78	86400	117683523	146529300	19.69	86400	134573452	148622727	9.45	86400
		2	32	131708441	162677081	19.04	86400	68361721	291103899	76.52	86400	114278868	138575597	17.53	86400	122530068	142860306	14.23	86400
		3	33	134440896	164789507	18.42	86400	67636653	343160408	80.29	86400	117928370	142211568	17.08	86400	120624839	145761305	17.24	86400
		4	34	127586963	149444104	14.63	86400	70982810	263425947	73.05	86400	127418133	133723286	4.72	86400	126525060	134672500	6.05	86400
		5	35	131406233	154859503	15.14	86400	68382521	300545433	77.25	86400	121555749	138769270	12.40	86400	114022733	139500252	18.26	86400
large	300	1	36	-	-	-	-	70035499	646987945	89.18	86400	156852657	222572311	29.53	86400	161746601	222572311	27.33	86400
		2	37	-	-	-	-	70035499	666721274	89.50	86400	157316568	225288927	30.17	86400	161424550	222941685	27.59	86400
		3	38	-	-	-	-	70035499	673069066	89.59	86400	158578879	227477765	30.29	86401	162174576	227477765	28.71	86400
		4	39	-	-	-	-	70035499	616335438	88.64	86400	154966468	213017148	27.25	86402	158330943	213017148	25.67	86400
		5	40	-	-	-	-	70035499	663794720	89.45	86400	160566612	221304838	27.45	86403	163573446	221327257	26.09	86400

**Table 4.7: Computational results for 10-time period**

Problem size	Nodes	Instance	Problem number	Hybridized SA					PSO							
				Obj value	D <sub>Gurobi-hSA</sub> (%)	D <sub>CBDD-hSA</sub> (%)	D <sub>ABDD-hSA</sub> (%)	D <sub>DBDD-hSA</sub> (%)	CPU Time	Avg Obj value	D <sub>Gurobi-PSO</sub> (%)	D <sub>CBDD-PSO</sub> (%)	D <sub>ABDD-PSO</sub> (%)	D <sub>DBDD-PSO</sub> (%)	D <sub>hSA-PSO</sub> (%)	CPU Time
(1)	(2)	(3)	(4)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)	(31)	(32)	(33)
small	50	1	21	35845574	0.02	-1.49	0.02	-0.60	1692	35844384	0.02	-1.49	0.02	-0.61	0.00	25609
		2	22	36032883	0.10	-0.44	-0.09	-0.52	4300	35996895	0.00	-0.54	-0.19	-0.62	-0.10	22827
		3	23	36840773	0.17	-1.05	0.01	-0.05	4753	36783064	0.01	-1.20	-0.14	-0.21	-0.16	14915
		4	24	34446146	0.00	-1.57	-2.09	-0.15	246	34446146	0.00	-1.57	-2.09	-0.15	0.00	3831
		5	25	35832720	0.04	-2.79	0.04	-0.01	2156	35822877	0.01	-2.82	0.01	-0.03	-0.03	28497
medium	100	1	26	69866988	2.37	-15.75	1.98	1.53	4674	70984701	4.01	-14.40	3.62	3.16	1.60	26426
		2	27	66325521	0.83	-17.18	0.17	-2.26	14690	69091579	5.04	-13.73	4.34	1.82	4.17	33429
		3	28	73121554	1.66	-21.13	0.68	-3.84	8480	74763040	3.94	-19.36	2.94	-1.69	2.24	22623
		4	29	64986637	0.67	-19.99	-0.42	-0.48	8665	64644368	0.14	-20.41	-0.94	-1.01	-0.53	26417
		5	30	66598021	1.04	-20.23	0.55	-2.02	9083	66019019	0.16	-20.92	-0.32	-2.87	-0.87	33871
medium	200	1	31	145896480	-11.07	-51.12	-0.43	-1.83	3839	156741589	-4.46	-47.49	6.97	5.46	7.43	37008
		2	32	142958749	-12.12	-50.89	3.16	0.07	3871	149236889	-8.26	-48.73	7.69	4.46	4.39	34796
		3	33	145991876	-11.41	-57.46	2.66	0.16	4046	159233707	-3.37	-53.60	11.97	9.24	9.07	31216
		4	34	139204640	-6.85	-47.16	4.10	3.37	4113	151482371	1.36	-42.50	13.28	12.48	8.82	36832
		5	35	142874545	-7.74	-52.46	2.96	2.42	4274	150728317	-2.67	-49.85	8.62	8.05	5.50	39821
large	300	1	36	213882500	-	-66.94	-3.90	-3.90	18000	247238389	-	-61.79	11.08	11.08	15.60	43200
		2	37	221619462	-	-66.76	-1.63	-0.59	7290	253024694	-	-62.05	12.31	13.49	14.17	30698
		3	38	209009804	-	-68.95	-8.12	-8.12	7221	279247752	-	-58.51	22.76	22.76	33.61	41196
		4	39	209009804	-	-66.09	-1.88	-1.88	7208	249119593	-	-59.58	16.95	16.95	19.19	33242
		5	40	216453004	-	-67.39	-2.19	-2.20	18000	246302946	-	-62.89	11.30	11.28	13.79	37958

**Table 4.8: Computational results for 15-time period**

Problem size	Nodes	Instance	Problem number	Gurobi				Classical Benders				Accelerated Benders				Benders Type Heuristic			
				LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)
small	50	1	41	55195583	55195583	0.00	61440	42015940	56423600	25.53	86400	52720424	55725979	5.39	86400	52913208	55574093	4.79	86400
		2	42	54975714	54975714	0.00	11051	44015826	55767968	21.07	86400	52135383	55195711	5.54	86400	54516446	55152480	1.15	86400
		3	43	55971310	55971310	0.00	65681	40000178	59145572	32.37	86400	53743812	56165161	4.31	86400	54259474	56300253	3.62	86400
		4	44	53289593	53289593	0.00	2462	43037026	54307237	20.75	86400	51483739	53289593	3.39	86400	52778633	53649620	1.62	86400
		5	45	55107888	55107888	0.00	42049	41506206	57107741	27.32	86400	53166346	55403651	4.04	86400	53371778	55234731	3.37	86400
medium	100	1	46	99931974	103614761	3.55	86400	64060948	145670194	56.02	86400	99350002	104451117	4.88	86400	101245831	109560010	7.59	86400
		2	47	96957242	100964928	3.97	86400	66040834	136587491	51.65	86400	96502266	101835183	5.24	86400	97757267	101479671	3.67	86400
		3	48	105104117	109765634	4.25	86400	64035948	154644255	58.59	86400	105243591	110860356	5.07	86400	106396856	110466536	3.68	86400
		4	49	95963329	99951084	3.99	86400	59534088	133376144	55.36	86400	96747904	101543821	4.72	86400	96909676	100033008	3.12	86400
		5	50	97555131	100963199	3.38	86400	60038622	136099953	55.89	86400	96692442	101361223	4.61	86400	97942889	101216894	3.23	86400
large	200	1	51	-	-	-	-	104070470	523401500	80.12	86400	170844018	221670528	22.93	86400	174191622	217710677	19.99	86400
		3	53	-	-	-	-	96563018	596117041	83.80	86400	165860254	218962474	24.25	86400	166945574	218964495	23.76	86400
		4	54	-	-	-	-	95042435	544542263	82.55	86400	171208856	203554532	15.89	86400	174228931	204576476	14.83	86400
		5	55	-	-	-	-	92560692	587746959	84.25	86400	167179400	216394562	22.74	86400	170261876	216413941	21.33	86400
large	300	1	56	-	-	-	-	116136904	1044180075	88.88	86400	229007622	325466525	29.64	86400	233762410	325466525	28.18	86400
		2	57	-	-	-	-	116136904	1074198487	89.19	86400	231130467	332713637	30.53	86400	234624408	332713637	29.48	86400
		3	58	-	-	-	-	116136904	1084556305	89.29	86400	228921278	339591886	32.59	86400	236151600	340030920	30.55	86400
		4	59	-	-	-	-	116136904	997196340	88.35	86400	227551124	313820212	27.49	86400	230665766	313820212	26.50	86400
		5	60	-	-	-	-	116136904	1069597382	89.14	86400	231609793	334973269	30.86	86400	236478578	334973269	29.40	86400

**Table 4.9: Computational results for 15-time period**

Problem size	Nodes	Instance	Problem number	Hybridized SA						PSO						
				Obj value	D <sub>Gurobi-hSA</sub> (%)	D <sub>CBDF-hSA</sub> (%)	D <sub>ABDF-hSA</sub> (%)	D <sub>BDh-hSA</sub> (%)	CPU Time	Avg Obj value	D <sub>Gurobi-PSO</sub> (%)	D <sub>CBDF-PSO</sub> (%)	D <sub>ABDF-PSO</sub> (%)	D <sub>BDh-PSO</sub> (%)	D <sub>hSA-PSO</sub> (%)	CPU Time
(1)	(2)	(3)	(4)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)	(31)	(32)	(33)
small	50	1	41	55238729	0.08	-2.10	-0.87	-0.60	11254	55549702	0.64	-1.55	-0.32	-0.04	0.56	31901
		2	42	54985252	0.02	-1.40	-0.38	-0.30	5699	55089813	0.21	-1.22	-0.19	-0.11	0.19	13339
		3	43	55584586	-0.69	-6.02	-1.03	-1.27	11254	56519351	0.98	-4.44	0.63	0.39	1.68	31715
		4	44	53289593	0.00	-1.87	0.00	-0.67	1100	53289593	0.00	-1.87	0.00	-0.67	0.00	19762
		5	45	55231400	0.22	-3.29	-0.31	-0.01	13419	55478462	0.67	-2.85	0.14	0.44	0.45	30087
medium	100	1	46	105464724	1.79	-27.60	0.97	-3.74	6719	111478247	7.59	-23.47	6.73	1.75	5.70	26075
		2	47	105230751	4.23	-22.96	3.33	3.70	3610	109591244	8.54	-19.76	7.62	7.99	4.14	25867
		3	48	115320544	5.06	-25.43	4.02	4.39	3222	118272940	7.75	-23.52	6.69	7.07	2.56	17334
		4	49	105193210	5.24	-21.13	3.59	5.16	4457	106287768	6.34	-20.31	4.67	6.25	1.04	28360
		5	50	105834353	4.82	-22.24	4.41	4.56	7095	109116736	8.08	-19.83	7.65	7.80	3.10	24772
large	200	1	51	215743152	-	-58.78	-2.67	-0.90	10800	229945387	-	-56.07	3.73	5.62	6.58	37130
		3	53	215384502	-	-63.87	-1.63	-1.63	10810	221760583	-	-61.44	4.97	4.97	6.71	37611
		4	54	203782113	-	-62.58	0.11	-0.39	10828	229846308	-	-58.09	12.12	11.56	11.99	30345
		5	55	211419174	-	-64.03	-2.30	-2.31	10809	228220520	-	-61.90	3.50	3.49	5.93	34729
		large	300	1	56	339460912	-	-67.49	4.30	4.30	18026	223959841	-	-63.52	17.04	17.04
2	57			349924000	-	-67.42	5.17	5.17	18000	380928624	-	-63.71	17.18	17.18	11.41	39560
3	58			345152479	-	-68.18	1.64	1.51	18070	389866549	-	-63.36	17.01	16.86	15.12	40017
4	59			318369126	-	-68.07	1.45	1.45	18072	397354039	-	-61.23	23.21	23.21	21.45	43200
5	60			351695216	-	-67.12	4.99	4.99	18000	386653001	-	-62.82	18.71	18.71	13.06	43200

**Table 4.10: Computational results for 20-time period**

Problem size	Nodes	Instance	Problem number	Gurobi				Classical Benders				Accelerated Benders				Benders Type Heuristic			
				LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time	LB	UB	Gap	CPU Time
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)
small	50	1	61	76881125	76881125	0.00	67274	54637818	84112030	35.04	86395	73545003	77207271	4.74	86400	75017450	77401522	3.08	86400
		2	62	76530950	76530950	0.00	6961	59793268	80531305	25.75	86396	73156832	76597895	4.49	86400	75175038	76903081	2.25	86400
		3	63	78344393	78344393	0.00	69286	54903863	85319073	35.65	86397	75478178	78547600	3.91	86400	77227998	78379560	1.47	86400
		4	64	75258187	75258187	0.00	8556	60412543	75921015	20.43	86398	72330319	75258187	3.89	86400	72460937	75258187	3.72	86400
		5	65	76871419	76871419	0.00	47203	55786568	82026890	31.99	86399	74073121	76871419	3.64	86400	75176220	76916363	2.26	86400
medium	100	1	66	141558877	148005946	4.36	86400	92129187	214040698	56.96	86400	140842632	148523960	5.17	86400	142036318	148675327	4.47	86400
		2	67	137924861	145837921	5.43	86400	92617705	213189420	56.56	86400	137335495	143502272	4.30	86400	138523051	144155473	3.91	86400
		3	68	148236698	155207481	4.49	86400	90936184	222026744	59.04	86400	148085430	155617894	4.84	86400	149151119	155515351	4.09	86400
		4	69	136351248	142081078	4.03	86400	85140887	196566695	56.69	86400	135550470	141771979	4.39	86400	137353247	142067172	3.32	86400
		5	70	138210751	142839915	3.24	86400	88603729	199472688	55.58	86400	137593763	142992805	3.78	86400	138940938	143431501	3.13	86400
large	200	1	71	-	-	-	-	141877427	845654087	83.22	86400	230376294	314066328	26.65	86400	234377768	314066328	25.37	86400
		2	72	-	-	-	-	132030124	872676074	84.87	86400	222000149	304315194	27.05	86400	230029742	304315194	24.41	86400
		3	73	-	-	-	-	135476742	855701438	84.17	86400	226533290	310645504	27.08	86400	237414808	310645504	23.57	86400
		4	74	-	-	-	-	132757389	827489641	83.96	86400	230571990	297526245	22.50	86400	240080024	299347856	19.80	86400
		5	75	-	-	-	-	128796035	870407916	85.20	86400	226028205	302820571	25.36	86400	234147612	302820571	22.68	86400
large	300	1	76	-	-	-	-	172226738	1495870337	88.49	86401	317845276	461958312	31.20	86400	320167678	462152076	30.72	86400
		2	77	-	-	-	-	172226738	1536681141	88.79	86402	321720866	463957617	30.66	86400	327306211	463957617	29.45	86400
		3	78	-	-	-	-	172226738	1551387856	88.90	86403	323731484	474096621	31.72	86400	329703191	474096621	30.46	86400
		4	79	-	-	-	-	172226738	1431977195	87.97	86404	318354612	443474416	28.21	86400	320832694	443474416	27.65	86400
		5	80	-	-	-	-	172226738	1530390960	88.75	86405	323907982	461184592	29.77	86400	329648546	465705409	29.22	86400



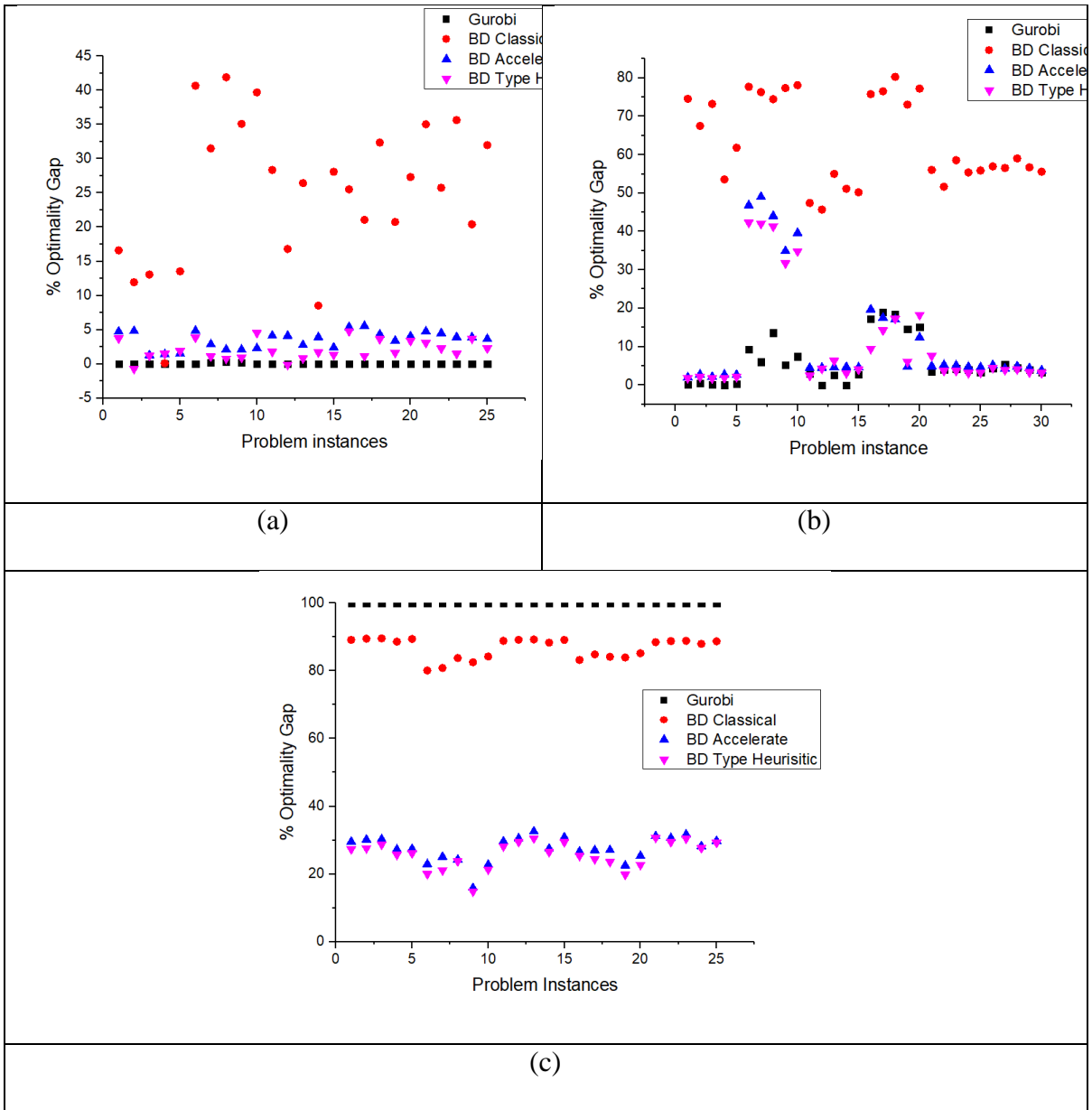
**Table 4.11: Computational results for 20-time period**

Problem size	Nodes	Instance	Problem number	Hybridized SA						PSO						
				Obj value	D <sub>Gurobi-hSA</sub> (%)	D <sub>CBD-hSA</sub> (%)	D <sub>ABD-hSA</sub> (%)	D <sub>BDb-hSA</sub> (%)	CPU Time	Avg Obj value	D <sub>Gurobi-PSO</sub> (%)	D <sub>CBD-PSO</sub> (%)	D <sub>ABD-PSO</sub> (%)	D <sub>BDb-PSO</sub> (%)	D <sub>hSA-PSO</sub> (%)	CPU Time
(1)	(2)	(3)	(4)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)	(31)	(32)	(33)
small	50	1	61	77173301	0.38	-8.25	-0.04	-0.29	9833	77388665	0.66	-7.99	0.23	-0.02	0.28	28744
		2	62	76584906	0.07	-4.90	-0.02	-0.41	39426	76989609	0.60	-4.40	0.51	0.11	0.53	26767
		3	63	78819256	0.61	-7.62	0.35	0.56	43200	78850855	0.65	-7.58	0.39	0.60	0.04	25449
		4	64	75301914	0.06	-0.82	0.06	0.06	30414	75414318	0.21	-0.67	0.21	0.21	0.15	27923
		5	65	76917763	0.06	-6.23	0.06	0.00	41376	77566047	0.90	-5.44	0.90	0.84	0.84	28662
medium	100	1	66	155654529	5.17	-27.28	4.80	4.69	13002	162291995	9.65	-24.18	9.27	9.16	4.26	33024
		2	67	154596967	6.01	-27.48	7.73	7.24	16597	156436384	7.27	-26.62	9.01	8.52	1.19	27188
		3	68	166291126	7.14	-25.10	6.86	6.93	19640	172343562	11.04	-22.38	10.75	10.82	3.64	24113
		4	69	150639252	6.02	-23.36	6.25	6.03	20148	153691188	8.17	-21.81	8.41	8.18	2.03	37091
		5	70	147418791	3.21	-26.10	3.10	2.78	12890	163386356	14.38	-18.09	14.26	13.91	10.83	17194
large	200	1	71	301082409	-	-64.40	-4.13	-4.13	18085	351675710	-	-58.41	11.97	11.97	16.80	41920
		2	72	295159166	-	-66.18	-3.01	-3.01	18029	331010889	-	-62.07	8.77	8.77	12.15	40802
		3	73	303819181	-	-64.49	-2.20	-2.20	18069	330913555	-	-61.33	6.52	6.52	8.92	34495
		4	74	286726180	-	-65.35	-3.63	-4.22	18104	323550874	-	-60.90	8.75	8.09	12.84	30859
		5	75	300886942	-	-65.43	-0.64	-0.64	18024	323850231	-	-62.79	6.94	6.94	7.63	36435
large	300	1	76	481101826	-	-67.84	4.14	4.10	18043	-	-	-	-	-	-	-
		2	77	484663886	-	-68.46	4.46	4.46	18116	-	-	-	-	-	-	-
		3	78	513697224	-	-66.89	8.35	8.35	18060	-	-	-	-	-	-	-
		4	79	471608308	-	-67.07	6.34	6.34	18199	-	-	-	-	-	-	-
		5	80	486335745	-	-68.22	5.45	4.43	18063	-	-	-	-	-	-	-

Out of the total 80 problem instances Gurobi solver could obtain optimal solutions for 24 instances and feasible (non-optimal) solutions for 31 instances. In the case of the remaining 25 problem instances, Gurobi could not even produce a feasible solution within the imposed time limit. The majority of these problem instances are of large size. As expected, the optimal solutions are obtained only for small size problem instances, whereas the feasible but not optimal solutions are observed mostly for medium-size instances. This clearly shows that the computational requirement increases significantly with the problem size in taking up the classical optimization approach.

In contrast to the direct use of MHCFL formulation using Gurobi, all three Benders decomposition approaches provide feasible solutions for all problem instances within the imposed CPU time limit. Figure 4.13 depicts the optimality gap that is the difference between UB and LB reported by all the three Benders based approaches. It can be seen that due to the time-consuming iterations and complex nature of the MP and DSP, the classical BD algorithm results in extremely high optimality gaps even for small-size problem instances. In this case, the average optimality gap obtained within the specified CPU time limit of 24 hours for small, medium, and large size problem instances are 25.13%, 63.64%, and 86.68%, respectively. On the other hand, Gurobi reports an average optimality gap of 0.03% within 7.7 hours and 7.00% within 24 hours for small and medium-size problem instances, respectively. Looking into the high optimality gap reported by classical BD algorithm, it is reasonable to conclude that classical BD algorithm underperforms Gurobi for small and medium-sized problems. However, for large-size problem instances, it at least provides a UB for the problem within the specified time limit whereas Gurobi solver fails to do so. The preceding observations

substantiate the role of some acceleration strategies that have been earlier proposed for speeding up iterations and eventually to result in a better bound for the BD algorithm.



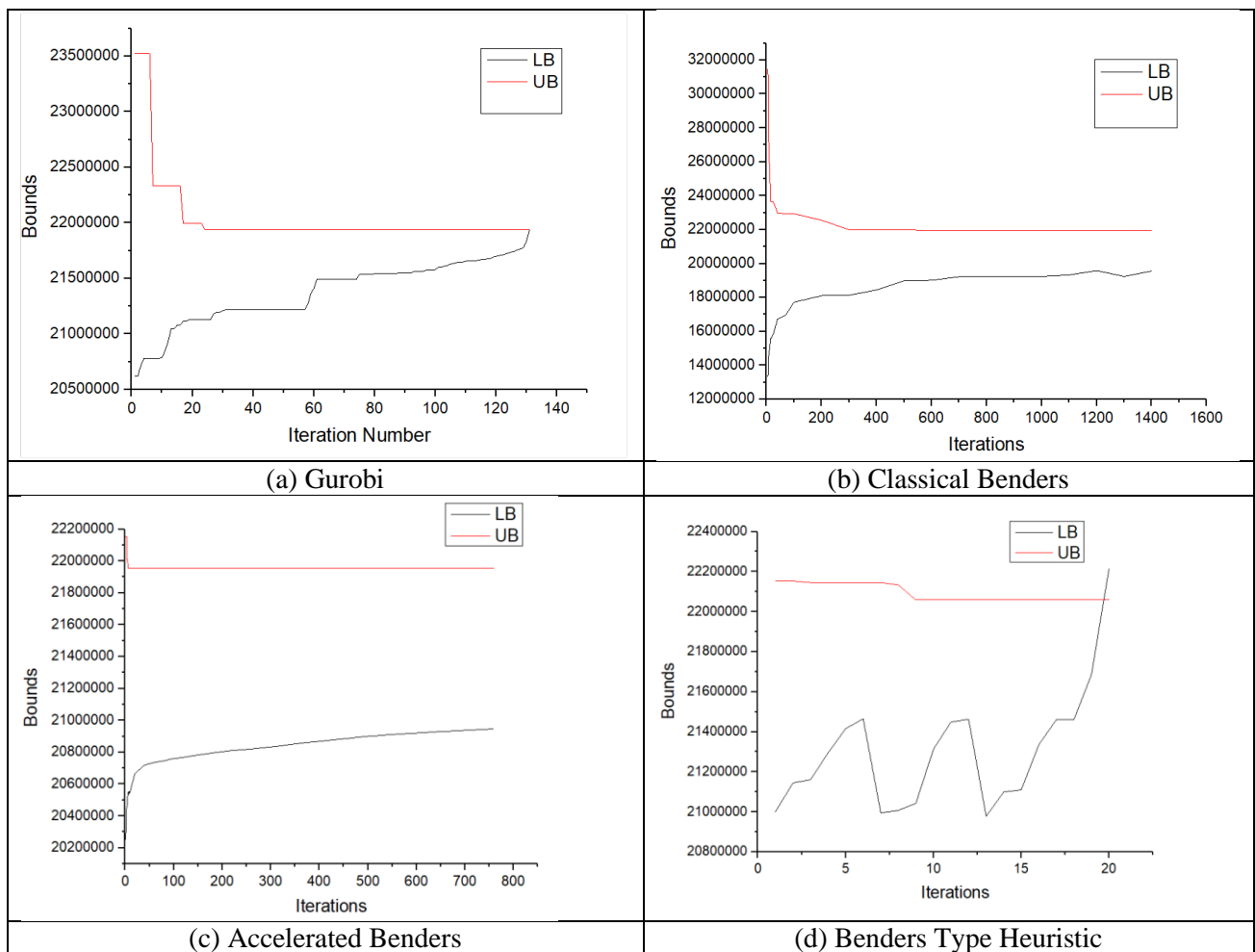
**Figure 4.13: Optimality gap obtained for (a) small, (b) medium and (c) large size problem instances**

Within 24 hours of the CPU time limit, the accelerated BD algorithm reported an average optimality gap of 3.54 % and 12.26 % for small and medium-size problem instances, respectively. It can be seen that these solutions are not superior to those from Gurobi solver in

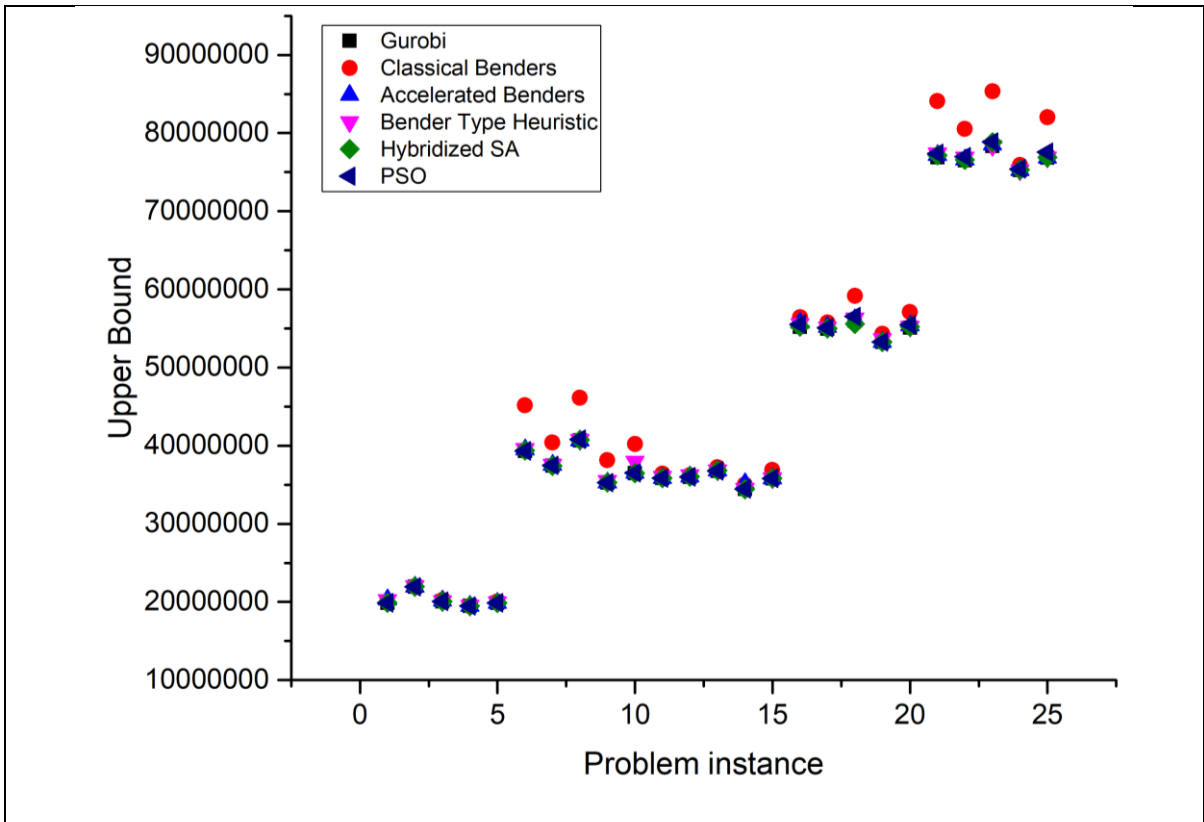
the case of small and medium instances. They, however, are comparable to Gurobi's solution and significantly better than that from the use of classical BD algorithm. In the case of large-size problem instances, the average optimality gap is 27.47 % in the use accelerated BD algorithm, producing as well a definitive bound on the solution obtained. The results are slightly improved when using the Benders type heuristic. Within the specified CPU time limit of 24 hours, the average optimality gap for small, medium, and large size problem instances in the case of Benders type heuristic are 2.06 %, 10.92 %, and 25.75 %, respectively. In some of the small size problem instances (problem number 2, 8, 22 and 23), the Benders type heuristic even produced an optimality gap of less than 1%. Thus, the results of the Benders type heuristic are far superior to those obtained by classical BD algorithm and are comparable to those obtained by accelerated BDA and Gurobi. It is evident that the use of LB improvement strategies such as relaxed Benders cut and parallelism have increased the lower bound up across all the problem classes. However, even after employing various acceleration strategies, both Benders decomposition based approaches struggle with convergence. A representative picture of the convergence reported by all the three Benders decomposition approaches and also by Gurobi is shown in Figure 4.14.

Although the optimality gap obtained by accelerated BD algorithm and Benders type heuristic is not superior to Gurobi, it can be seen that both Benders decomposition based strategies outperform Gurobi solver if the comparison is made based on upper bounds. Figure 4.15 depicts a comparison based on UB obtained from all the proposed approaches. Here, the comparison is made considering the UB reported by Gurobi as the base. For small and medium-sized problem instances, the average deviation of the UB obtained through accelerated BDA is 0.35% and -1.11%, respectively. This means that the UB obtained through the accelerated

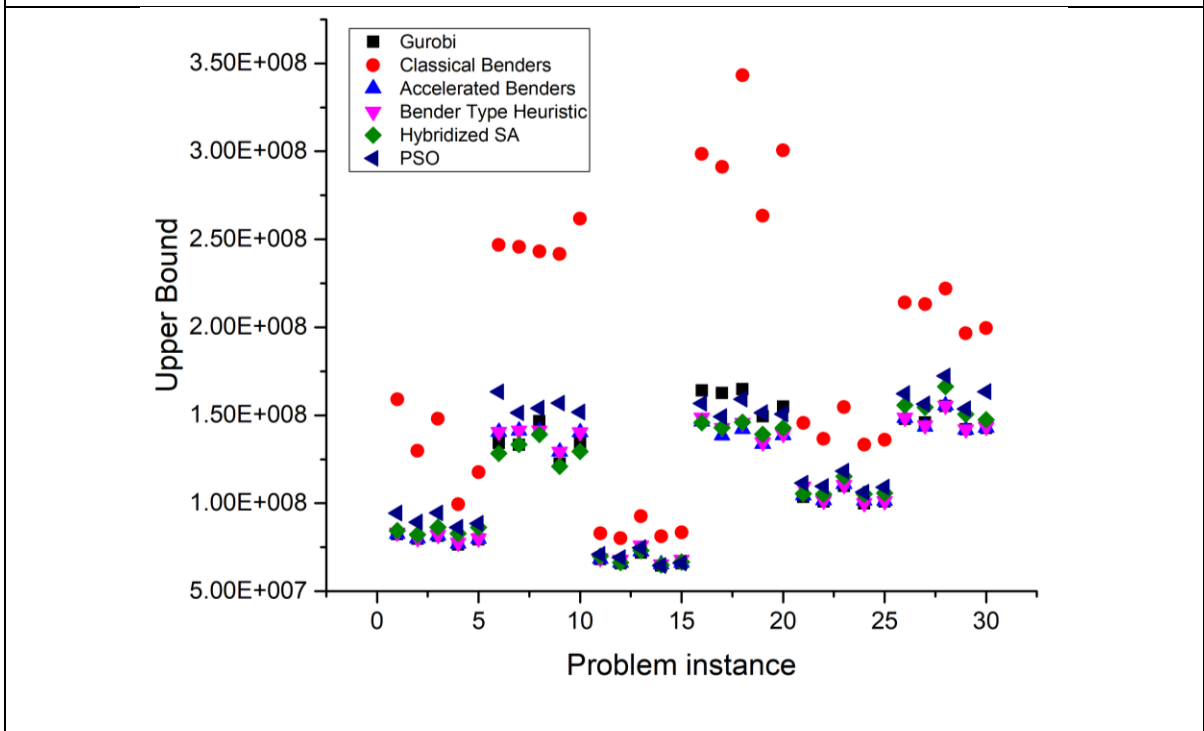
BDA is only 0.35% higher than that from Gurobi solver for small instances, and it is better than that for Gurobi solver for medium instances on an average. A deviation of less than 1% is observed in 23 out of 25 small size problems and 25 out of 30 medium-size problems. A similar pattern is observed in the case of the Benders type heuristic. For small and medium-sized problem instances, the average deviation in UB from Gurobi solver and Benders type heuristic is 0.58 % and -0.34 %, respectively. In this case, less than 1% deviation is observed in 23 out of 25 small size problems and 20 out of 30 medium-size problems.



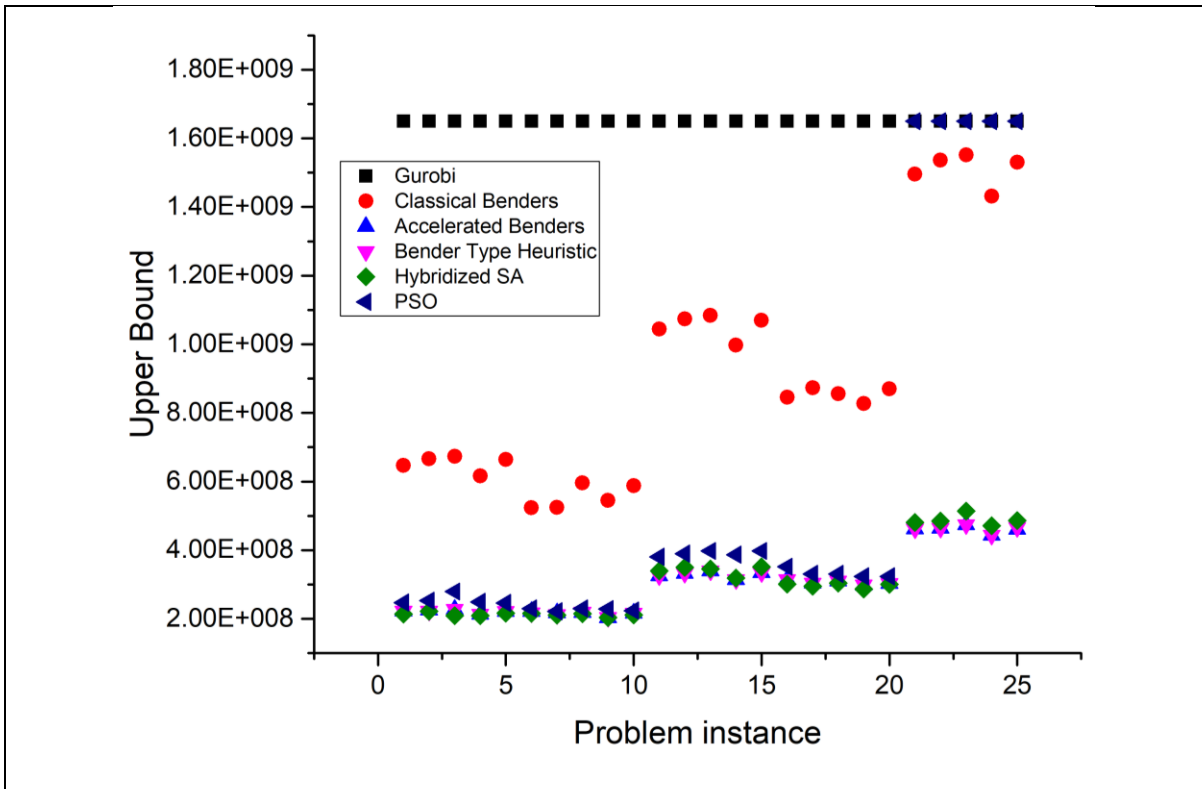
**Figure 4.14: Convergence of a problem of 50 nodes 5 time period and 2<sup>nd</sup> instance**



(a)



(b)



(c)  
**Figure 4.15: Comparison of UB obtained for (a) small, (b) medium (c) large size problem instances**

In terms of CPU time, it can be seen that Gurobi solver is efficient as compared to the other approaches. It produces an optimal solution for small size problem instances in 7.7 hours on an average basis. It took 24 hours of time for almost all the instances to solve the problem. In contrast, all Benders decomposition based approaches are run for 24 hours, and so they are not as efficient as Gurobi solver. It is interesting to note that for medium-size problems, the accelerated BD algorithm and Benders type heuristic reported better UB the Gurobi solver within the same CPU time limit of 24 hours.

The use of hybridized SA is experienced to be found better in most of the problem instances compared to Gurobi. For small size problem instances, hybridized SA yielded a near-optimal solution for 19 problem instances out of 25, and for 2 problem instances, it provides better UB

than that from Gurobi solver. For the medium size problem instances, , hybridized SA reported better solutions in 9 problems (out of 30) than from Gurobi solver. The maximum deviation is 8.84%. The hybridized SA also provide a good feasible solution for large size problem instances in very less CPU time. It is worth mentioning that even the maximum CPU time limit of 12 hours is not touched by hybridized SA for 76 problem instances. The comparison of UB obtained from all the proposed approaches is shown in Figure 4.15. Furthermore, looking at the resulting objective function values for the problem instances, it is observed that hybridized SA outperform the other three Benders decomposition approaches. However, for some medium and large size problem instances, the Benders type heuristic outperformed hybridized SA in terms of UB.

The objective value obtained from PSO is now compared with all other approaches discussed above, and the percentage deviation is shown in Table 4.5, 4.7, 4.9 and 4.11. When the objective function value of the solution obtained from PSO is compared with the UB from Gurobi solver, it is found that the deviation is less than 1% in all the small size problem instances. For the medium size problem instances, mixed results are obtained. For 5 time-period problems, the deviation is higher, whereas as the problem size increases (for medium and large size problems), the deviation improves to the better side. PSO provides a good quality solution in a specified time limit of 12 hours for large-scale problems whereas Gurobi solver fails to report any feasible solution. PSO approach has also failed to report feasible solutions for 5 large-scale problem instances. PSO performed better compared to the classical Benders decomposition, but fell short compared with accelerated Benders decomposition and Benders type heuristic. The solutions from PSO are also compared with the solutions from hybridized SA. It is found that, hybridized SA reported better solutions than PSO in 70 problem instances.



In many cases, the deviation and reported CPU time are very close in both the cases. So to have a clear and exact conclusion on the comparative performance of PSO and hybridized SA, Wilcoxon Signed-Rank tests were conducted. The test found the hybridized SA to be better than PSO in terms of solution quality ( $p = 0.00001$ ) and CPU time ( $p = 0.00001$ ).

An analysis of the CPU time used by all six approaches shows that the direct use of MHCFL formulation using a solver is not advisable compared to the other five approaches except for when it results in the optimal solution. It can further be observed that out of the three variants of Benders decomposition, classical BD algorithm performed worst, whereas Benders type heuristic and accelerated BD algorithm performed comparatively better in terms of solution quality. PSO performed well compared to Gurobi solver but fell short when compared to other approaches in terms of both solution quality and CPU time. The hybridized SA approach is found to be the most efficient and effective for several problem instances.

#### **4.9 Integrated vs Disaggregated Planning**

In the earlier chapters, the healthcare network plans were designed keeping a particular period in mind. The problem and the corresponding mathematical model did not consider the possibility of upgrading the facilities. The problem and the corresponding MHCFL model design the maternal healthcare facilities network plan considering the possibility of upgrading the existing facilities. The model proposed in the chapters established various facility types in the first period and considered the possibility of their upgradation in subsequent years. Thus the planning size for the second period will take into cognizance the healthcare facilities that are planned to be created in the first period. This possibility could not be captured by the mathematical model presented in the earlier chapters.

The earlier section of this chapter has shown that the multi-period problem involves too many design variables and constraints while formulating it as a mathematical programming model. To avoid this problem, one may think of solving the problem sequentially and period-wise. Though the computational complexity is expected to decrease, the quality of the solution in terms of the objective function value is likely to deteriorate. Therefore one needs to analyse how much beneficial the integrated consideration of the problem could be viz-a-viz when they are solved sequentially. If the difference is not very large, one may like to follow the sequential approach. To analyze this aspect from the managerial perspective, a detailed what-if analysis has been carried out.

The experimentation has been carried out on a 50 node problem for a planning horizon of 5, 10, 15 and 20 years. The results of this analysis have been presented in Tables 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19 and 4.20. The results shown in these tables exhibit that the sequential approach asks for a lesser number of facilities to be established by permitting degradation in service quality (high penalty cost and more overburdening). At the same time, the sequential approach absorbs more of the cost. The cost considered here is the inclusion of penalty cost. On making penalty costs very high for 20-time period problem, it is found that overburdening becomes zero. (for this purpose, please refer to Tables 4.19 and 4.20). To avoid overburdening, it is observed that facilities are upgraded at a later stage besides establishing more of the facilities. The results presented in Table 4.21 clearly show that integrated consideration is advantageous compared to the sequential approach. The amount of advantage increases with the increase in the number of periods in the planning horizon.

**Table 4.12: Solution of MHCFL formulation for 5-time period**

Time period	Number of new			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC	
1	3	0	0	1	0	1	6	4	3	0	0	206	<b>1.99E+07</b>
2	0	0	0	0	0	0	6	4	3	0	0	220	
3	0	0	0	0	0	0	6	4	3	0	0	232	
4	0	0	0	0	0	0	6	4	3	0	0	245	
5	0	0	0	0	0	0	6	4	3	0	0	258	

**Table 4.13: Solution of sequential approach for 5-time period**

Time period	Number of new			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value	Total objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC		
1	2	0	0	0	0	1	6	3	3	315	631	207	8.11E+06	<b>2.38E+07</b>
2	0	0	0	0	0	0	6	3	3	491	714	221	3.43E+06	
3	0	0	0	0	0	0	6	3	3	659	797	232	3.75E+06	
4	0	0	0	0	0	0	6	3	3	828	880	245	4.07E+06	
5	0	0	0	0	0	0	6	3	3	1000	972	260	4.41E+06	

**Table 4.14: Solution of MHCFL formulation for 10-time period**

Time period	Number of new			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC	
1	4	0	0	1	0	1	7	4	7	0	0	207	<b>3.58E+07</b>
2	0	0	0	0	0	0	7	4	7	0	0	221	
3	0	0	0	0	0	0	7	4	7	0	0	232	
4	0	0	0	0	0	0	7	4	7	0	0	245	
5	0	0	0	0	0	0	7	4	7	0	0	260	
6	0	0	0	0	0	0	7	4	7	0	55	271	
7	0	0	0	0	0	0	7	4	7	0	145	283	
8	0	0	0	0	0	0	7	4	7	0	230	298	
9	0	0	0	0	0	0	7	4	7	0	318	313	
10	0	0	0	0	0	0	7	4	7	0	416	326	

**Table 4.15: Solution of sequential approach for 10-time period**

Time period	Number of new			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value	Total objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC		
1	2	0	0	0	0	1	6	3	3	315	631	207	8.11E+06	<b>5.10E+07</b>
2	0	0	0	0	0	0	6	3	3	491	714	221	3.43E+06	
3	0	0	0	0	0	0	6	3	3	659	797	232	3.75E+06	
4	0	0	0	0	0	0	6	3	3	828	880	245	4.07E+06	
5	0	0	0	0	0	0	6	3	3	1000	972	260	4.41E+06	
6	0	0	0	0	0	0	6	3	3	1174	1053	271	4.74E+06	
7	0	0	0	0	0	0	6	3	3	1349	1143	283	5.09E+06	
8	0	0	0	0	0	0	6	3	3	1525	1229	297	5.43E+06	
9	0	0	0	0	0	0	6	3	3	1704	1318	311	5.78E+06	
10	0	0	0	0	0	0	6	3	3	1887	1415	326	6.15E+06	

**Table 4.16: Solution of MHCFL formulation for 15-time period**

Time period	Number of new			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC	
1	5	0	0	1	1	0	7	5	3	0	0	207	<b>5.52E+07</b>
2	0	0	0	0	0	0	7	5	3	0	0	221	
3	0	0	0	0	0	0	7	5	3	0	0	232	
4	0	0	0	0	0	0	7	5	3	0	0	245	
5	0	0	0	0	0	0	7	5	3	0	0	259	
6	0	0	0	0	0	0	7	5	3	0	0	271	
7	0	0	0	0	0	0	7	5	3	0	0	283	
8	0	0	0	0	0	0	7	5	3	0	0	298	
9	0	0	0	0	0	0	7	5	3	0	0	313	
10	0	0	0	0	0	0	7	5	3	0	0	326	
11	0	0	0	0	0	0	7	5	3	0	0	339	
12	0	0	0	0	0	0	7	5	3	0	0	351	
13	0	0	0	0	0	0	7	5	3	0	0	366	
14	0	0	0	0	0	0	7	5	3	0	0	379	
15	0	0	0	0	0	0	7	5	3	0	0	392	

**Table 4.17: Solution of sequential approach for 15-time period**

Time period	Number of new			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value	Total objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC		
1	2	0	0	0	0	1	6	3	3	315	631	207	8.11E+06	<b>8.73E+07</b>
2	0	0	0	0	0	0	6	3	3	491	714	221	3.43E+06	
3	0	0	0	0	0	0	6	3	3	659	797	232	3.75E+06	
4	0	0	0	0	0	0	6	3	3	828	880	245	4.07E+06	
5	0	0	0	0	0	0	6	3	3	1000	972	260	4.41E+06	
6	0	0	0	0	0	0	6	3	3	1174	1053	271	4.74E+06	
7	0	0	0	0	0	0	6	3	3	1349	1143	283	5.09E+06	
8	0	0	0	0	0	0	6	3	3	1525	1229	297	5.43E+06	
9	0	0	0	0	0	0	6	3	3	1704	1318	311	5.78E+06	
10	0	0	0	0	0	0	6	3	3	1887	1415	326	6.15E+06	
11	0	0	0	0	0	0	6	3	3	2072	1503	337	6.51E+06	
12	0	0	0	0	0	0	6	3	3	2258	1599	351	6.89E+06	
13	0	0	0	0	0	0	6	3	3	2445	1691	365	7.27E+06	
14	0	0	0	0	0	0	6	3	3	2633	1784	378	7.65E+06	
15	0	0	0	0	0	0	6	3	3	2822	1880	391	8.04E+06	

**Table 4.18: Solution of MHCFL formulation for 20-time period**

Time period	Number of new			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC	
1	5	0	0	0	1	1	8	3	4	0	0	0	<b>8.11E+07</b>
2	0	0	0	0	1	1	8	3	4	0	0	0	
3	0	0	0	0	1	1	8	3	4	0	0	0	
4	0	0	0	0	1	1	8	3	4	0	0	0	
5	0	0	0	0	1	1	8	3	4	0	0	0	
6	0	0	0	0	1	1	8	3	4	0	0	0	
7	0	0	0	0	1	1	8	3	4	0	0	0	
8	0	0	0	0	1	1	8	3	4	0	31	0	
9	0	0	0	0	1	1	8	3	4	0	118	13	
10	0	0	0	0	1	1	8	3	4	0	216	26	
11	0	0	0	0	1	1	8	3	4	0	303	40	
12	0	0	0	0	1	1	8	3	4	0	400	51	
13	0	0	0	0	1	1	8	3	4	0	492	67	
14	0	0	0	0	1	1	8	3	4	0	585	80	
15	0	0	0	0	1	1	8	3	4	0	682	93	
16	0	0	0	0	1	1	8	3	4	0	773	106	
17	0	0	0	0	1	1	8	3	4	0	873	121	
18	0	0	0	0	1	1	8	3	4	0	967	135	
19	0	0	0	0	1	1	8	3	4	92	1065	150	
20	0	0	0	0	1	1	8	3	4	291	1167	164	

**Table 4.19: Solution of sequential approach for 20-time period**

Time period	Number of new			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value	Total objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC		
1	2	0	0	0	0	1	6	3	3	315	631	207	8.11E+06	<b>1.34E+08</b>
2	0	0	0	0	0	0	6	3	3	491	714	221	3.43E+06	
3	0	0	0	0	0	0	6	3	3	659	797	232	3.75E+06	
4	0	0	0	0	0	0	6	3	3	828	880	245	4.07E+06	
5	0	0	0	0	0	0	6	3	3	1000	972	260	4.41E+06	
6	0	0	0	0	0	0	6	3	3	1174	1053	271	4.74E+06	
7	0	0	0	0	0	0	6	3	3	1349	1143	283	5.09E+06	
8	0	0	0	0	0	0	6	3	3	1525	1229	297	5.43E+06	
9	0	0	0	0	0	0	6	3	3	1704	1318	311	5.78E+06	
10	0	0	0	0	0	0	6	3	3	1887	1415	326	6.15E+06	
11	0	0	0	0	0	0	6	3	3	2072	1503	337	6.51E+06	
12	0	0	0	0	0	0	6	3	3	2258	1599	351	6.89E+06	
13	0	0	0	0	0	0	6	3	3	2445	1691	365	7.27E+06	
14	0	0	0	0	0	0	6	3	3	2633	1784	378	7.65E+06	
15	0	0	0	0	0	0	6	3	3	2822	1880	391	8.04E+06	
16	0	0	0	0	0	0	6	3	3	3012	1972	404	8.43E+06	
17	0	0	0	0	0	0	6	3	3	3203	2072	420	8.83E+06	
18	0	0	0	0	0	0	6	3	3	3396	2166	433	9.23E+06	
19	0	0	0	0	0	0	6	3	3	3592	2263	448	9.65E+06	
20	0	0	0	0	0	0	6	3	3	3791	2364	462	1.01E+07	

**Table 4.20: Solution of sequential approach for 20-time period with very high penalty**

Time period	Number of New			Number of facilities upgraded from			Number of facilities operating			Overburdening			Objective function value	Total Objective function value
	SC	PHC	CHC	SC to PHC	SC to CHC	PHC to CHC	SC	PHC	CHC	BU	MH	NC		
1	2	0	0	1	0	2	5	3	4	0	0	0	1.22E+07	8.42E+07
2	0	0	0	0	0	0	5	3	4	0	0	0	2.24E+06	
3	1	0	0	0	0	0	6	3	4	0	0	0	3.34E+06	
4	0	0	0	0	0	0	6	3	4	0	0	0	2.34E+06	
5	0	0	0	0	0	0	6	3	4	0	0	0	2.42E+06	
6	0	0	0	0	0	0	6	3	4	0	0	0	2.51E+06	
7	0	0	0	0	0	0	6	3	4	0	0	0	2.61E+06	
8	0	0	0	1	0	0	5	4	4	0	0	0	5.40E+06	
9	0	0	0	0	0	1	5	3	5	0	0	0	7.11E+06	
10	0	0	0	0	0	0	5	3	5	0	0	0	3.12E+06	
11	1	0	0	0	0	0	6	3	5	0	0	0	4.67E+06	
12	0	0	0	0	0	0	6	3	5	0	0	0	3.31E+06	
13	0	0	0	0	0	0	6	3	5	0	0	0	3.44E+06	
14	0	0	0	0	0	0	6	3	5	0	0	0	3.56E+06	
15	0	0	0	0	0	0	6	3	5	0	0	0	3.70E+06	
16	1	0	0	0	0	0	7	3	5	0	0	0	5.60E+06	
17	0	0	0	0	0	0	7	3	5	0	0	0	3.94E+06	
18	0	0	0	0	0	0	7	3	5	0	0	0	4.10E+06	
19	0	0	0	0	0	0	7	3	5	0	0	0	4.25E+06	
20	0	0	0	0	0	0	7	3	5	0	0	0	4.42E+06	

**Table 4.21: Comparison of the objective value of MHCFL formulation and sequential approach**

Time period	MHCFL	Sequential	Difference (%)
5	19884302	23777925.93	19.58
10	35837986	50974534.9	42.23
15	55195583	87333533.38	58.22
20	81064799	133534090.8	64.72

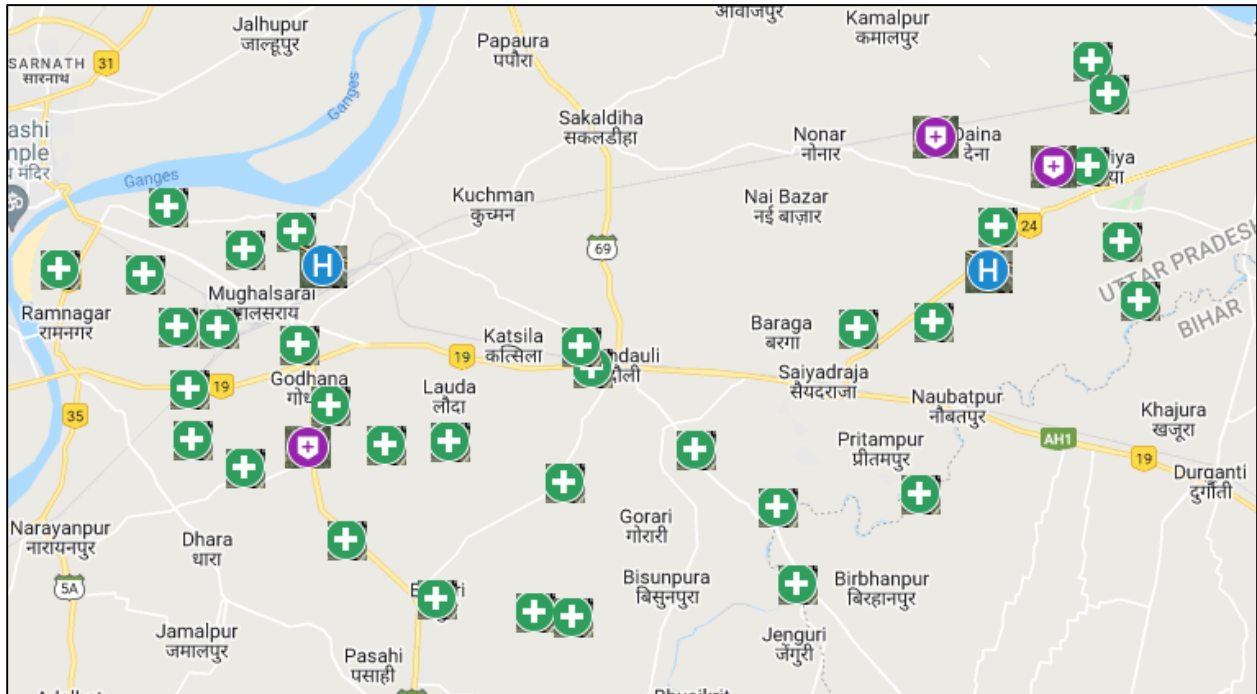
#### **4.10 Case example**

The problem of healthcare facility location and allocation for the district of Chandauli in Uttar Pradesh, India, has been discussed in this section. According to the 2011 census, Chandauli has a population of 19,52,756, with a population density of 769 people per square kilometre (Chandauli, 2011). It is divided into three Tehsils (administrative units at the next level): Chakia, Chandauli, and Sakaldiha constituting 624, 522, and 494 villages, respectively (Chandauli, 2021). The problem of Chandauli Tehsil has been taken as the case problem. The population of the village is taken from census statistics (Chandauli, 2011). Several of the villages were found to have a population being less than 100 people. Because of the number of MTBs from such villages being extremely small, the population of these villages is combined with the population of the nearby village. As a result, the number of significant villages was reduced to 356. The villages' coordinates (latitude and longitude) were obtained using the Google search engine, and the road distance between them was determined using the Bing Maps API (Sinani, 2013). The maximum coverage distance for a normal visit is taken as 5 kilometres, while for the referral visits as 7 kilometres (Rural Health Statistics, 2019). According to the Pradhan Mantri Surakshit Matritva Abhiyan's scheme, the number of MTBs in a village is calculated by multiplying the population size and the birth rate in that village (PMSMA, 2016).

MTBs at each location were considered to be distributed uniformly throughout the year. The population of the district is expected to grow at a rate of 1% per year (Economic survey, 2019). The referral proportions are the same as in Section 4.5.1. Table 4.22 shows values for other factors, including capacity, fixed cost, up-gradation cost, and operational cost. The cost of establishment, upgradation, and operating the health facilities are considered to increase at a



rate of 4% per period due to inflation. The number of doctors, staff nurses, ward boys, and other paramedical staff available in each facility type determines the capacity for each service type and is kept the same for the facilities of the same type.



**Figure 4.16: Locations of the different types of existing facilities**

**Legends:**

Facilities at existing locations		SCs		PHC		CHC
----------------------------------	---	-----	---	-----	---	-----

All the solution approaches outlined in Section 4.4 are used to solve the problem. Tables 4.23 and 4.24 display these results. Table 4.23 finds hybridized SA to yield the best solution. The CPU time taken is small and marginally high compared to the least CPU time taken by PSO. The performance of Benders decomposition technique, used in any form, is moderate. Table 4.24 shows establishment of 8 SCs, 59 PHCs, and 48 CHCs in the first year of the planning horizon. In Chandauli Tehsil, there are 72 SCs, 5 PHCs, and 3 CHCs, presently existing (Figure 4.16) and all of these are not solely devoted for MTBs. Existence of 5 PHCs against 59 PHCs,

and of 3 CHCs against required 48 CHCs shows a lot of significant improvement required in the Indian healthcare system. It should be noted that the comparison is on general healthcare to the maternal healthcare. From this perspective, the gap is definitely very wide. It is for this reason, it is being emphasised that the existing facilities should be solely devoted to healthcare facilities other than that for MTBs. For MTBs, the plan be made for creation of separate facilities as desired under SDG of UN.

The problem of improper mix and shortage of healthcare services in the Tehsil is obvious from the solution. Figure 4.17 shows the location of existing facilities along with those locations where new facilities are to be established during the planning horizon. Figure 4.17 also shows the locations and maximum area covered by the CHCs. Table 4.24 shows the number of facilities of each type required over the next 20 years. The capacity of various facility types find the final number of CHCs and SCs in the first year to be sufficient to meet the demand even at a growth rate of 1%. However, PHCs are required to be increased time to time in response to the increased demand. The plan resulted for maternal healthcare is perfect to deal with the issue of accessibility and availability in the Chandauli Tehsil for the next 20 years. The solution takes care of the service quality and also provides projection for financial outlay.

**Table 4.22: Problem parameters used in the case problem**

Parameters	Facility type		
	SC	PHC	CHC
Capacity			
Service type 1	300	600	1200
Service type 2	0	240	500
Service type 3	0	0	300
Establishment cost	2000000	20000000	60000000
Upgradation cost	SC	18000000	58000000
	PHC	-	40000000
Operating cost	3000000	9000000	27000000

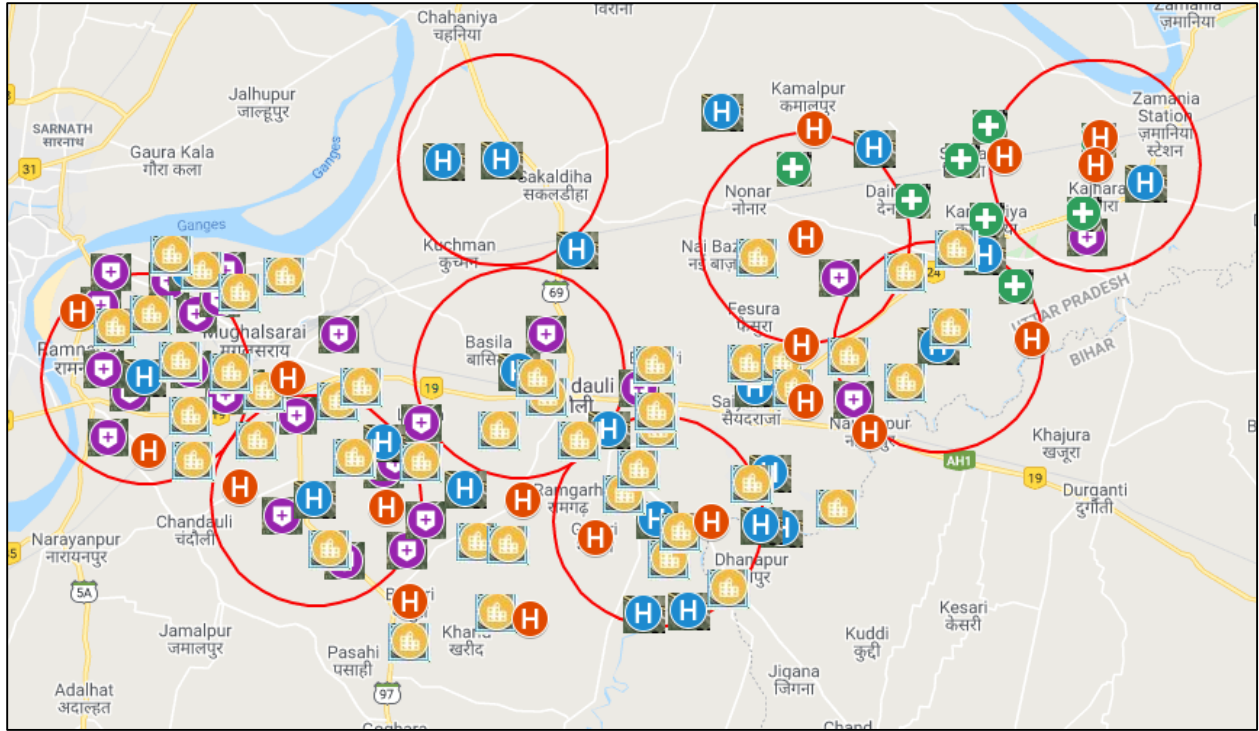
*\*each data is in units*

**Table 4.23: Objective function value obtained from proposed solution approaches**

	<b>LB</b>	<b>UB</b>	<b>Gap (%)</b>	<b>CPU Time (in sec)</b>
<b>MHCFL</b>	62385848552	62520386450	0.22	43200
<b>Classical BD algorithm</b>	60284532159	69856235103	13.70	43200
<b>Accelerated BD algorithm</b>	62324136580	62935412602	0.97	43200
<b>Benders Type Heuristic</b>	6192450364	629214893301	90.16	43200
<b>Hybridized SA</b>	-	62518738687	-	19723
<b>PSO</b>	-	64765705620	-	19680








**Table 4.24: Number of different types of facilities required**

<b>By the end of the Year</b>	<b>Facility type</b>		
	<b>SC</b>	<b>PHC</b>	<b>CHC</b>
1	8	59	48
2	8	59	48
3	8	59	48
4	8	60	48
5	8	61	48
6	8	62	48
7	8	63	48
8	8	63	48
9	8	64	48
10	8	66	48
11	8	67	48
12	8	69	48
13	8	71	48
14	8	73	48
15	8	75	48
16	8	75	48
17	8	75	48
18	8	75	48
19	8	75	48
20	8	75	48
Initial distribution	72	5	3



**Figure 4.17: Locations of the different types of facilities**

**Legends:**

<b>Facilities at new locations</b>		<b>CHCs,</b>		<b>PHCs,</b>		<b>SCs</b>
<b>Facilities at existing locations</b>		<b>SCs</b>		<b>SCs</b>		<b>PHCs</b>
				<b>Upgraded to CHC</b>		<b>Upgraded to CHC</b>

## 4.11 Conclusions

In this chapter, a mathematical model for addressing the multi-period facility location-allocation problem regarding maternal healthcare has been presented. The proposed model takes population growth into account and also considers the up-gradation of existing healthcare facilities to respond to the growing demand in various time periods of the planning horizon. The issue of accessibility has been addressed by explicitly considering the critical distance an MTB has to cover to reach a healthcare facility. The issue of availability and service quality is

handled by considering the facility's limited capacity and a penalty cost against overloading of the same. The objective also considered is the minimization of the total cost, which includes the cost of establishing a new facility, up-gradation cost, facility operating cost, and transportation costs. Because of the computational complexity involved in applying usual optimization approaches using solver, accelerated Benders decomposition algorithm, Benders type heuristic, hybridized simulated annealing, and particle swarm optimization have been proposed. Extensive computational experiments were carried out to compare the effectiveness and computational efficiency of the proposed solution approaches. On overall basis, accelerated BDA and Benders type heuristic outperformed the Gurobi solver, particle swarm optimization performed worst, while hybridized SA was found to be the best in terms of yielding quality solutions in less time. The analyses carried out shows that it is advantageous to consider the planning problem for a long planning horizon. Year to year basis planning is not an intelligent proposition in economic terms. The proposed mathematical model was utilized for developing the maternal healthcare facility network for the Tehsil of Chandauli in Uttar Pradesh, India, for the next 20 years. The resulting solution shows that the present scenario is incapable of providing quality service and needs urgent support for establishing exclusive maternal healthcare facilities.

