Regionalization of Primary Health Care Units: An Iterated Greedy Algorithm for Large-Scale Instances

Rodolfo Mendoza-Gómez [a]
Tecnológico de Monterrey
School of Engineering and Science
Eugenio Garza Sada SN, Cerro Gordo
León, Guanajuato 37190, Mexico
E-mail: *rodolfomendoza@tec.mx*

Roger Z. Ríos-Mercado
Universidad Autónoma de Nuevo León (UANL)
Graduate Program in Electrical Engineering
Av. Universidad s/n, Cd. Universitaria
San Nicolás de los Garza, Nuevo León 66455, Mexico
E-mail: *roger.rios@uanl.edu.mx*

September 2023
Revised: December 2023

[a]Corresponding author

Abstract

In this paper, we study the problem of multi-institutional regionalization of primary health care units. The problem consists of deciding where to place new facilities, capacity expansions for existing facilities, and demand allocation in a multi-institutional system to minimize the total travel distance from demand points to health care units. It is known that traditional exact methods as branch-and-bound are limited to solving small- to medium-size instances of the problem. Given that real world-instances can be large, in this paper we propose an iterated greedy algorithm with variable neighborhood descent search for handling large-scale instances. Within this solution framework, several methods are developed. A greedy constructive method and two deconstruction strategies are developed. Another interesting component is the exact optimization of a demand allocation subproblem that is obtained when the location of facilities is previously fixed. An empirical assessment using real-world data from the State of Mexico's Public Health Care System is carried out. The results demonstrate the effectiveness of the proposed metaheuristic in handling large-scale instances.

Keywords: Public health care planning; Facility location; Metaheuristics; Iterated greedy algorithm.

1 Introduction

Discrete facility location is an important area in operations research and computer science. There are many applications in industry and the public sector for a wide range of problems that include factories, warehouses, distribution centers, retailer stores, schools, police stations, health care units, ambulance stations, offices, and so on. An extensive recent survey of location models is provided by Laporte et al. [19], and Ahmadi-Javid et al. [3] provide an extensive survey of location models applied to health care. Some contributions related to the locational planning of health care units are proposed by Marianov and Taborga [22], Marianov et al. [23], Griffin et al. [16], Ndiaye and Alfares [30] , Smith et al. [36], Gu et al. [17], Shariff et al. [35], and de Aguiar et al. [8]. Others, such as Mitropoulos et al. [28], Zhang et al. [38] and Mendoza-Gómez and Ríos-Mercado [25], address multi-objective optimization problems. Furthermore, works integrating stochastic parameters are explored by Taymaz et al. [37] and Ahmadi-Javid and Ramshe [2].

In this paper, we are dealing with the problem of regionalization of primary health care units (HCUs) in a segmented public system proposed by Mendoza-Gómez and Ríos-Mercado [24]. We refer to this problem as the Multi-Institution Facility Location and Upgrading Problem (MIFLUP). The problem consists of determining the location of new capacity in the system. This can be done by opening new facilities or adding more capacity to the existing HCUs. The allocation of demand is required because the capacity of HCUs is limited. There is a set of institutions, and each institution has a demand to serve at each demand point, but when the capacity is not enough to fulfill the demand or there are no HCUs nearby, collaboration among institutions can be done to share the services. In this case, the allocation of demand to other institutions can be done, but this is constrained by a set of policies. The general objective of this problem is to minimize the total weighted distance from demand points to HCUs. The main goals are to improve the population's access to these facilities and to ensure a minimum quality level in the provision of primary health care services.

This problem can be seen as a variation of the capacitated p-median problem (CPMP) with additional side constraints. The objective of this problem is to find the optimal location of p facilities, considering distances and capacities for the service to be given by each median. The CPMP problem has been proven to be $\mathcal{NP}$-hard by a reduction from the p-median problem [13]. This means that optimal solutions can be difficult to obtain for larger instances of the problem using exact algorithms. The MIFLUP includes additional features as the capacity setting and the inter-institutional allocation, requiring us to design alternative solutions methods for large-scale instances.

Among the exact approaches that have been proposed for the CPMP, a branch-and-price algorithm that exploits column generation, heuristics, and branch-and-bound to compute optimal solutions for the CPMP is proposed by Ceselli and Righini [7]. In Boccia et al. [5], a cutting plane algorithm, based on Fenchel cuts, is used to reduce the integrality gap of hard CPMP instances. Related to heuristic methods, one of the first metaheuristics is proposed by Osman and Christofides [32]. They propose a hybrid simulated annealing and a Tabu search algorithm. Maniezzo et al. [21] propose a bionomic algorithm and a local search for the CPMP. Baldacci et al. [4] propose an exact algorithm based on a set partitioning formulation. Díaz and Fernández [9] combine Scatter Search and path relinking algorithms, using GRASP (Greedy Randomized Adaptive Search Procedure) to generate the initial reference set. Ahmadi and Osman [1] propose a new solution framework based on GRASP and adaptative memory programming. Then, a guided construction search metaheuristics is proposed by Osman and Ahmadi [31]. Recently, Gnägi and Baumann [14] propose a metaheuristic with decomposition strategies.

Work on facility location models on segmented health care systems has been done by Mendoza-Gómez et al. [26] and Mendoza-Gómez et al. [27]. They address the problem of locating specialized health care equipment in the Mexican Health Care System (MHCS). A hybrid metaheuristic based on the iterated greedy algorithm is proposed. In fact, that work has similarities with the present work that we attempt to exploit in the development of our solution procedure. A related model for HCUs applied to the MHCS is presented by Mendoza-Gómez and Ríos-Mercado [25]. In that work, one institution is considered in the system.

The problem addressed in this work is introduced by Mendoza-Gómez and Ríos-Mercado [24]. In that work, an integer programming model is proposed. Empirical evidence using branch-and-bound made clear the need for heuristics to handle large-scale instances. To the best of our knowledge, there are no heuristics developed for this particular problem. Since the objective is to obtain a practical decision that is yearly required for hundreds of regions in the country, a good quality solution obtained in a reasonable time can be used. This solution can be obtained with metaheuristics that are faster but give up optimality. In a practical setting, one approach is to solve smaller regional problems and then integrate this solution as a whole. However, it is clear that this may lead to suboptimal solutions when considering the nation-wide problem. An alternative is to consider the entire system which is intractable by exact algorithms. This motivates the development of heuristic techniques as proposed in this paper.

The main contribution of the paper is the development of a hybrid metaheuristic framework for tackling large-scale instances of this problem. Note that, to the best of our knowledge, there are no other heuristic methods for this particular problem. The proposed strategies make use of several components as an iterated greedy (IG) algorithm and variable neighborhood descent algorithm (VND), that attempt to exploit the mathematical structure of the problem through the exploration of two proposed neighborhoods. Some particular input parameters for the IG and the constructive method are proposed to reduce the problem's working space during the solution construction. In addition, within the solution procedures, we present an allocation subproblem that can be solved with an efficient exact method as a final step using the heuristic solutions as the starting feasible solution. Hence, this method represents a hybrid metaheuristic, featuring components that hold relevance for other trajectory-based metaheuristics.

IG is a simple but powerful metaheuristic framework, introduced by Ruiz and Stützle [34] for solving combinatorial optimization problems. IG is similar to GRASP proposed by Feo and Resende [11], but in this case, instead of randomizing the construction of a solution, it is partially randomly destroyed, and then, using a constructive strategy, the solution is rebuilt. VNS is a metaheuristic proposed by Mladenović and Hansen [29] that systematically modifies the structure of a set of neighborhoods in the search procedure. A specific simple strategy is to select the neighborhood in a deterministic order, this strategy is named the variable neighborhood descent search (VND). An implementation of this metaheuristic in a related problem is provided by Fleszar and Hindi [12] for the CPMP. There many recent works where the IG framework is used to solve complex problem such as Qin et al. [33], Hoffmann et al. [18], Feng et al. [10], Zou et al. [39], and Liu et al. [20] applied to scheduling problems, Casado et al. [6] for finding the minimum dominating set in graphs, and Gokalp [15] for the obnoxious p-median problem.

An empirical assessment applied to a case study of MCHS in the State of Mexico is conducted. Eighteen instances with a range between five hundred to three thousand demand points are used to evaluate different strategies and fine-tune parameters of the metaheuristic. The results show a good performance of the metaheuristic compared with the state-of-the-art branch-and-bound algorithm (B&B). While no feasible solutions were found for the largest instances by B&B with a two-hour computing time limit, the metaheuristic is able to find feasible solutions in all the cases, and competitive or even better solutions are found in most of the cases where B&B found feasible solutions.

The structure of this paper is as follows. In Section 2, we present the model introduced by Mendoza-Gómez and Ríos-Mercado [24] for a better understanding of the proposed heuristic components. In Section 3, we describe the proposed metaheuristic and all the related components and algorithmic strategies. Section 4 presents the results of the empirical assessment applied to a case study. Finally, conclusions of this work are drawn in Section 5.

2 Problem Description

MIFLUP was introduced by Mendoza-Gómez and Ríos-Mercado [24] and they proved that it is classified as an $\mathcal{NP}$-hard problem. The objective of this problem is to allocate demand points to capacitated HCUs of multiple institutions minimizing the total weighted travel distance. New facilities can be installed and new capacity can be added to the system if these options contribute to minimizing the objective function. A percentage of capacity at each HCU can be used to allocate the demand of other institutions. The capacity is based on a modular scheme named basic kernels. In this scheme, a kernel is composed of a physician, a nurse, and a technician in primary health care that can serve a limited number of inhabitants in the region. Figure 1 illustrates the problem considering three institutions, five demand points, three existing HCUs, and one candidate site to build a HCU. Each HCU has a given capacity and additional kernels can be installed on it. The candidate site (CS) can be considered a HCU but without installed capacity in the current system. A maximum number of new locations and a maximum number of new kernels are available for each institution. There is demand that belongs to each institution at each demand point. Therefore, the demand of each institution at each demand point must be assigned to a single HCU. The HCU may belong to the same institution or another if it has enough capacity available. The number of binary variables for each demand point is determined by the number of institutions and the number of total locations (HCUs and CSs) as can be seen in the variables related to demand point 1 in the figure.

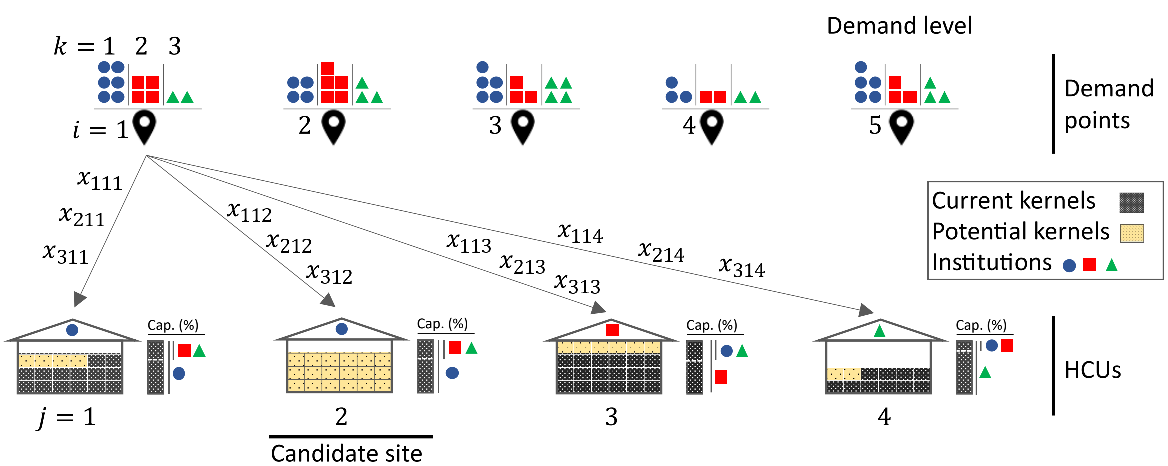

Figure 1: Graphical representation of the problem.

2.1 Formulation

For the sake of completeness and a better understanding of the proposed solution method, we present an integer linear programming formulation of the problem that was taken from the model proposed by Mendoza-Gómez and Ríos-Mercado [24]. In our specific case, we assume that there is no minimum capacity to be allocated in each HCU. Thus, constraints (5) from Mendoza-Gómez and Ríos-Mercado [24] become redundant. The notation, parameters, and variables used in the problem formulation are the following:

Sets and indices:

N Set of demand points ($i, l \in N$).

M Set of (existing and candidate) locations for HCUs ($t, j \in M$).

K Set of institutions $(k, r \in K)$.

M_k Set of locations (HCUs and CSs) that belong to institution k.

M_A Set of locations where a HCU is already installed, $M_A \subseteq M$.

M_B Set of CSs for installing a HCU, $M_B \subseteq M$.

M_{B_k} Set of CSs that belong to institution k.

$k(j)$ The institution to which location j belongs to, $j \in M_r \leftrightarrow r = k(j)$.

Parameters:

KC Kernel capacity. Maximum number of inhabitants covered by a kernel.

d_{ij} Distance from demand point i to HCU location j; $i \in N$, $j \in M$.

w_{ki} Demand of institution k in point i; $k \in K$, $i \in N$.

a_j Number of current kernels in HCU j. A value of a_j equal to zero indicates there is no current HCU at that place; therefore, it is a CS for installing a HCU; $j \in M$.

V_j Maximum number of kernels that can be installed in location j; $j \in M$.

H_j Minimum number of kernels that must be installed in location j if a HCU is opened; $j \in M_B$.

β_k Maximum proportion of capacity in a HCU of institution k that can be shared to the demand of other institutions; $k \in K$.

γ_k Minimum proportion of demand that institution k must be cover internally; $k \in K$.

G_k Maximum number of additional kernels of institution k that can be installed; $k \in K$.

P_k Number of new HCU to be opened by institution k; $k \in K$.

Decision variables:

x_{kij} $= 1$, if demand of institution k at demand point i is allocated to HCU j; $= 0$, otherwise; $k \in K$, $i \in N$, $j \in M$.

y_j $= 1$, if a HCU is opened at location j; $= 0$, otherwise; $j \in M_B$.

v_j Integer variable equal to the number of additional kernels to be opened in HCU j; $j \in M$.

The linear integer programming model of MIFLUP is then given by:

$$\text{Minimize} \quad f(x) = \sum_{i \in N} \sum_{j \in M} \sum_{k \in K} w_{ki} d_{ij} x_{kij} \tag{1}$$

$$\text{subject to:} \quad \sum_{j \in M} x_{kij} = 1 \qquad k \in K,\ i \in N \tag{2}$$

$$\sum_{r \in K: r \neq k(j)} x_{rij} \leq (|K| - 1) x_{kij} \qquad k \in K,\ i \in N,\ j \in M_k \tag{3}$$

$$\sum_{i \in N} \sum_{k \in K} w_{ki} x_{kij} \leq KC(a_j + v_j) \qquad j \in M \tag{4}$$

$$\sum_{i \in N} \sum_{r \in K: r \neq k(j)} w_{ri} x_{rij} \leq KC(a_j + v_j) \beta_{k(j)} \qquad j \in M \tag{5}$$

$$\sum_{i \in N} \sum_{j \in M_k} w_{ki} x_{kij} \geq \gamma_k \sum_{i \in N} \sum_{k \in K} w_{ki} \qquad k \in K \tag{6}$$

$$\sum_{j \in M_k} v_j \le G_k \qquad k \in K \tag{7}$$

$$x_{kij} \le y_j \qquad k \in K,\ i \in N,\ j \in M_B \tag{8}$$

$$v_j \ge H_j y_j \qquad j \in M_B \tag{9}$$

$$\sum_{j \in M_{B_k}} y_j \le P_k \qquad k \in K \tag{10}$$

$$v_j \le V_j \qquad j \in M_A \tag{11}$$

$$v_j \le V_j y_j \qquad j \in M_B \tag{12}$$

$$x_{kij} \in \{0, 1\} \qquad k \in K,\ i \in N,\ j \in M \tag{13}$$

$$y_j \in \{0, 1\} \qquad j \in M_B \tag{14}$$

$$v_j \in \mathbb{Z}^+ \qquad j \in M \tag{15}$$

The total travel distance is minimized in the objective function (1). Constraints (2) enforce allocating all the demand of each institution at each demand point to a single HCU or CS. Constraints (3) avoid allocating the demand of other institutions in a given demand point to a HCU or CS if the demand of the institution to which the HCU or CS belongs is not allocated first. Constraints(4) limit the demand allocation according to the capacity of each HCU or CS determined by the number of actual kernels plus new kernels. Constraints (5) limit the demand of other institutions allocated to each HCU or CS according to $\beta_{k(j)}$. Constraints (6) are used to guarantee a minimum percentage of demand allocated internally for each institution. Constraints (7) set the maximum number of new kernels that can be installed for each institution. Constraints (8) prevent allocating demand to a CS that is not selected. Constraints (9) define a minimum number of kernels to be installed in the selected CS. Constraints (10) set the maximum number of selected CS for each institution. Constraints (11) and (12) define the maximum number of kernels that can be installed in existing HCUs and CS, respectively. Finally, the nature of decision variables is defined in constraints (13) and (15).

3 Proposed metaheuristic

In this section, we describe an IG algorithm to solve MIFLUP that was introduced in Section 2. The main procedure is shown in Pseudo-code 1. Three components are used in the multi-start procedure: the deconstruction strategy, the constructive strategy, and a local search procedure. A solution is represented by $\mathcal{S}$ and the best feasible solution found is represented by $\mathcal{S}^*$. The objective function value of any feasible solution is represented by $Z(\mathcal{S})$. The procedure is iterated until a stopping criterion (*stopping_criterion*) is satisfied. This criterion can be for instance a computing time limit, a given number of iterations, or a combination of both criteria. The percentage of deconstruction is determined by parameter ρ, which is fine-tuned. Four additional input parameters (Φ, D_0, D_1, D_2) are required in the construction and deconstruction phases of this problem. D_1 and D_2 are used in the constructive and deconstruction strategies to reduce the number of operations in the procedure by reducing the size of the environment affected when a solution is modified. This is a special feature for large-scale instances. All these parameters are explained in the following subsections.

We present in this paper a constructive method (CM), two deconstruction strategies (DS1 and DS2), two types of neighborhoods that give rise to two local search schemes (LS1 and LS2), two versions of a VND (VND12 and VND21), and a sub-problem to be optimized (ALLOP) as elements that can be included or combined within the IG. In Table 1, we proposed some heuristics methods generated with this elements that are evaluated in Section 4. In the second column, the elements of the IG are represented as follows IG{*deconstructive strategy*, *constructive method*, *local search strategy*}. In H9 and H10, there is an extra final method that consists in optimizing the allocation subproblem (ALLOP) with an exact method. Figure 2 show a representative diagram of the entire framework used for design the proposed heuristics. In the following subsections, all the components proposed are explained in detail.

Pseudocode 1 Iterated greedy algorithm

```
procedure IG(stopping_criterion, ρ, Φ, D0, D1, D2)
    S* ← INITIALIZATION_PROCEDURE(D0);
    S* ← CM(S*, Φ, D1);
    while stop_criteria is not satisfied do
        S ← S*;
        S ← DM(S, ρ, D2);
        S ← CM(S, Φ, D1);
        S ← LOCAL_SEARCH(S);
        if (Z(S) < Z(S*)) then
            S* ← S;
        end if
    end while
return (S*)
end procedure
```

Table 1: Description of the proposed heuristics.

ID	Heuristic method	Description
H1	IG{DS1, CM, – }	use DS1 at each iteration.
H2	IG{DS2, CM, – }	use DS2 at each iteration.
H3	IG{DS1+DS2, CM, – }	use DS1 and DS2 sequentially at each iteration.
H4	IG{DS2+DS1, CM, – }	use DS2 and DS1 sequentially at each iteration.
H5	IG{DS1+DS2, CM, LS1+LS2}	use H3 and apply LS1 and LS2 sequentially at each iteration.
H6	IG{DS1+DS2, CM, LS2+LS1}	use H3 and apply LS1 and LS2 sequentially at each iteration.
H7	IG{DS1+DS2, CM, VND12 }	use H3 and apply VND12 at each iteration.
H8	IG{DS1+DS2, CM, VND21 }	use H3 and apply VND21 at each iteration.
H9	IG{DS1+DS2, CM, VND12 }+ALLOP	use H7 and optimize ALLOP.
H10	IG{DS1+DS2, CM, VND21 }+ALLOP	use H8 and optimize ALLOP.

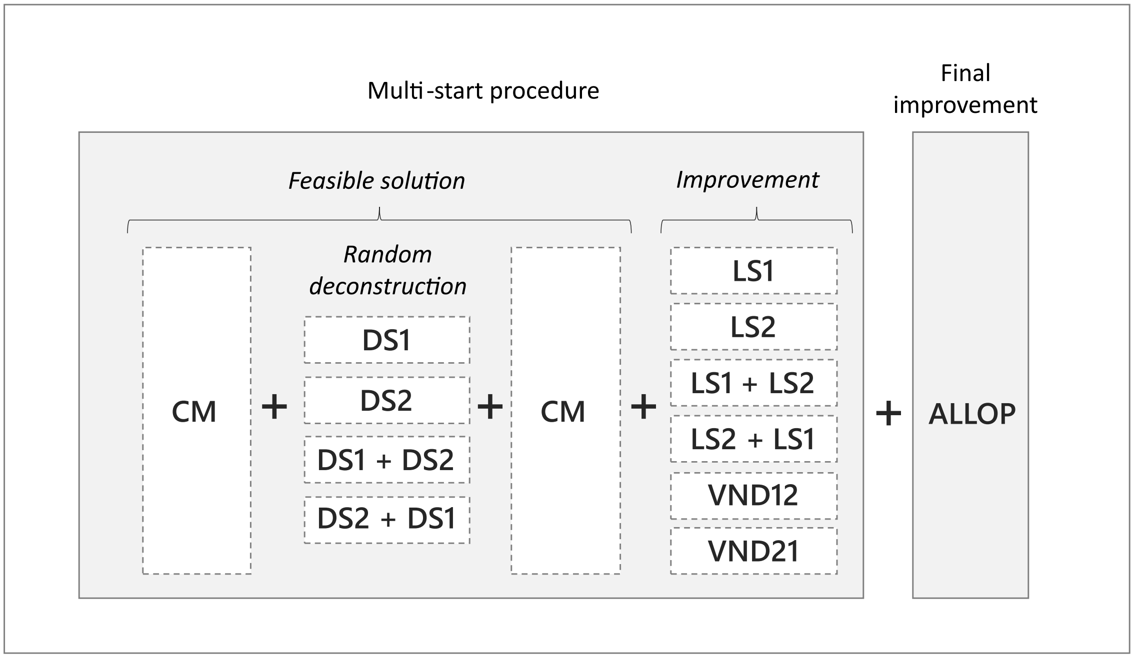

Figure 2: Framework for the heuristics design.

Representation of a solution

A solution $\mathcal{S}$ is composed of three main variable types that are initialized and then modified throughout the algorithm. They are defined as follows:

$\tilde{y}_j$ is a variable associated with y_j for all $j \in M_B$. It stores a value equal to 1 if a HCU is opened at CS j and 0 otherwise.

$\tilde{v}_j$ is a variable associated with v_j for $j \in M$. It represents the number of additional kernels installed in j.

$\tilde{x}_{ki}$ is a variable associated with x_{kij} for $k \in K,\ i \in N,\ j \in M$. This variable stores the site j such that $x_{kij} = 1$. This change helps to reduce the computing memory required to store solutions.

A given solution is represented by $(\tilde{y}, \tilde{v}, \tilde{x})$ or $\mathcal{S}$. However, there are many working parameters that are useful for identifying the residual capacity and the solution's feasibility when the procedure is running: $\tilde{w}_{ki}$, C_j, CI_j, O_k, $\tilde{V}_j$, $\tilde{G}_k$, $\tilde{P}_k$, and λ_{ki}. These working parameters can be computed from $\mathcal{S}$ or updated every time there is a change in the solution. Their definition is available in the Appendix and they are used in all the procedures that are described below. The decision variables and the working parameters must be initialized at the beginning of the IG as is shown in Subsection 3.1.

Figure 3 shows a small example to illustrate the terminology that is used to describe the procedures. In this example, the elements belong to institution $k = 1$. In the case of HCUs ($j \in M_{A_1}$), some of them can increase their capacity with additional kernels ($\tilde{v}_j \geq 0$). In the case of candidate sites ($j \in M_{B_1}$), some of them may be selected to open new HCUs with new kernels ($\tilde{y}_j = 1$ and $\tilde{v}_j > 0$) and others can not be selected ($\tilde{y}_j = 0$ and $\tilde{v}_j = 0$). For the last two cases, we use the terms "selected candidate sites (SCS)" and "unselected candidate sites (UCS)", respectively. For the case of HCUs and SCS, we refer to these elements as "active sites" because represent existing and new HCUs. In this problem, we require to identify where to install new kernels ($\tilde{v}_j$) and how to allocate the demand of each institution ($\tilde{x}_{ki}$). For the case of kernels, we use the terms "assign and unassign" kernels to a site j and for the case of demand, we use the terms "allocate and deallocate" demand to a site j.

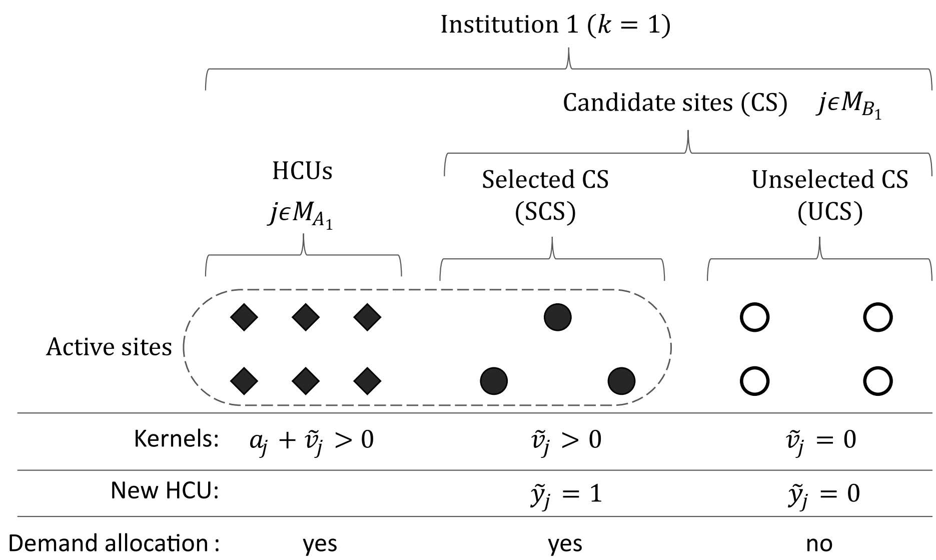

Figure 3: Example of the terminology of the elements in the solution.

3.1 Initialization procedure

Pseudo-code 2 shows the initial value for the solution ($\tilde{y}_j, \tilde{v}_j, \tilde{x}_{ki}$) and the working parameters ($\tilde{w}_{ki}$, C_j, CI_j, O_k, $\tilde{V}_j$, $\tilde{G}_k$, $\tilde{P}_k$, and λ_{ki}). This subroutine is required for the constructive method when there is no partial or complete solution created. The working parameters must be updated every time there is a change in the solution. In Step 2, the working parameters are initialized. In Step 3, λ_{ki} is initialized with a distance D_0 and when a demand point is allocated to an active site, the parameter is updated according to Equation 26. The purpose of this parameter is to find an active site with a better distance than D_0. The decision variables are created with initial values in Step 4 that represents an unfeasible solution. The constructive method can be applied once this initialization is done as is shown in Pseudocode 1.

Pseudocode 2 Initialization

procedure INITIALIZATION_PROCEDURE(D_0)
  Compute the working parameters as follows:
  $$\tilde{w}_{ki} = w_{ki}\ \forall(k \in K, i \in N),\ \ C_j = KC(a_j)\ \forall j \in M_A,\quad C_j = 0\ \forall j \in M_B,$$
  $$CI_j = \beta_{k(j)} C_j\ \forall j \in M,\quad O_k = (1-\gamma_k)\textstyle\sum_{i \in N} w_{ki}\ \forall k \in K,\ \ \tilde{V}_j = V_j\ \forall j \in M;$$
  $$\tilde{G}_k = G_k\ \forall k \in K,\ \ \tilde{P}_k = P_k\ \forall k \in K;$$
  Compute λ_{ki} according to (26) and D_0;
  Compute the solution as follows:
  $$\tilde{y}_j = 0\ \forall j \in M_B,\quad \tilde{v}_j = 0\ \forall j \in M,\quad \tilde{x}_{ki} = \infty\ \forall(k \in K, i \in N);$$
  $\mathcal{S} \leftarrow (\tilde{y}, \tilde{v}, \tilde{x})$;
return ($\mathcal{S}$)
end procedure

3.2 Constructive method (CM)

The main steps of CM are shown in Pseudo-code 3. In this procedure, an initial solution $\mathcal{S}$ and two input parameters: Φ and D_1, are required as input. Parameter Φ defines the strategy for determining the number of new kernels to add to each site as follows: option (i) computes the minimum possible number of kernels, option (ii) computes the average value between the minimum and the maximum possible number of kernels, and option (iii) computes the maximum possible number of kernels. Parameter D_1 defines the influence area for sites whose capacity is modified in the procedure.

In Step 2, for each institution k, the best P_k candidate sites are selected, new kernels are assigned to these SCS without exceeding G_k, and demand is allocated to them according to the available capacity. Then, in Step 3, available kernels ($\tilde{G}_k$) for each institution are assigned to strategic active sites, and demand is allocated to them. In Step 4, the remaining demand is allocated to active sites with available capacity ($C_j > 0$). If there is the case that some candidate sites can still be selected ($\sum_{k \in K} \tilde{P}_k > 0$), this is forced in Step 6, selecting additional candidate sites and transferring kernels to them. The output of this algorithm provides a greedy feasible solution for the entire problem. CM can be used to complete a partial solution in the reconstruction phase of the IG. In the following subsections, each of these procedures is detailed explained.

Pseudocode 3 Constructive algorithm

1: **procedure** CM($\mathcal{S}$, Φ, D_1)
2:   $\mathcal{S} \leftarrow$ OPEN_NEW_FACILITIES($\mathcal{S}$, Φ);
3:   $\mathcal{S} \leftarrow$ OPEN_NEW_KERNELS($\mathcal{S}$, Φ);
4:   $\tilde{x} \leftarrow$ DEMAND_ALLOCATION($\mathcal{S}$);
5:   **if** ($\sum_{k \in K} \tilde{P}_k > 0$) **then**
6:     $\mathcal{S} \leftarrow$ KERNELS_TRANSFERS($\mathcal{S}$, D_1);
7:   **end if**
8: **return** ($\mathcal{S}$)
9: **end procedure**

3.2.1 CM: Open new facilities

Procedure OPEN_NEW_FACILITIES() is shown in Pseudo-code 4. The objective is to select candidate sites to install new HCUs for each institution according to P_k. A candidate list CL with UCS of all institutions is created in Step 2. Then, in Step 4, the function (δ_j), the potential new capacity (PC_j), and the number of new kernels ($u_j^1(\Phi)$) are computed for each element according to Equations (27), (30), and (31), respectively. Parameter δ_j computes the improvement in the allocation distance of each demand point to the candidate site multiplied by the demand rate. The procedure ends when neither of the elements generates a benefit ($\sum \delta_j > 0$) or when CL is empty. Otherwise, the element with the highest δ_j value is selected to become an active site (Step 6). The working parameters associated with the residual capacity are updated in Steps 7 and 8, and the solution is updated in Step 9.

For allocating demand is required to call a subroutine DEMAND_ASSIGNMENT() that is shown in Pseudo-code 5. This subroutine allocates demand to the active site t considering the incumbent solution. Firstly, a list of demand points (k, i) with specific criteria is created and stored in DPL according to Steps 2–8. In this list, demand points (k, i) with a distance to site t such that $d_{it} < \lambda_{ki}$ are considered. Parameter λ_{ki} represents the current allocation distance of each demand point (k, i). Therefore, only demand points such that the allocation distance can be improved are considered. The demand $\tilde{w}_{ki}$ must not exceed the available capacity of site t. In the case of inter-institutional allocation ($k \neq k(t)$), the feasibility of constraints (3), (5), and (6) must be also fulfilled (Step 6). This last requirement is evaluated with the following conditions: $\tilde{x}_{k(t)i} = t$, $\tilde{w}_{ki} \leq CI_t$, $\tilde{w}_{ki} \leq O_k$. For each demand point in the DPL, θ_{ki} is computed to determine the benefit of this demand point if this is allocated to the active site t. One by one, demand points are allocated to this active site starting from the ones with the highest values. The working parameters related to the residual capacity are updated in Step 12 and the solution is updated in Step 13. The DPL list of demand points is updated with the same criteria as the Steps 2–8 and the procedure is repeated until the DPL becomes empty. In this case, only $\tilde{x}$ is returned as output because the other variables were not modified.

3.2.2 CM: Addition of new kernels

Procedure OPEN_NEW_KERNELS() shown in Pseudo-code 6 is called to assign kernels, according to $\tilde{G}_k$ for each institution. A candidate list CL of sites such that $\tilde{V}_j > 0$ is created. The number of kernels to assign ($u_j^2(\Phi)$), PC_j, and δ_j are computed with Equations (28), (30), and (31), respectively. Then, the procedure is very similar to Pseudo-code 4. In this case, $u_j^2(\Phi)$ is used to compute the potential capacity to be added for each element of CL. Other changes are the working parameters that are updated in Step 8 and the solution in Step 9. This process is repeated while $\sum \tilde{G}_k > 0$ or $CL \neq \{\emptyset\}$. At the end of each iteration, CL is updated with the same criteria as Step

Pseudocode 4 Open new facilities

procedure OPEN_NEW_FACILITIES($\mathcal{S}$, Φ)
  $CL \leftarrow \{j \in M_B | \tilde{y}_j = 0,\ \tilde{P}_{k(j)} > 0,\ \tilde{G}_{k(j)} \geq H_j\}$;
  while $(CL \neq \emptyset)$ **do**
    compute $u_j^1(\Phi)$, PC_j, and δ_j $\forall j \in CL$ according to (27), (30), and (31);
    if $(\sum \delta_j \neq 0)$ **then**
      $t \leftarrow \arg\max_{j \in CL}\{\delta_j\}$;
      $C_t \leftarrow C_t + PC_t, \quad \tilde{V}_t \leftarrow \tilde{V}_t - u_t^1(\Phi), \quad CI_t \leftarrow CI_t + \beta_{k(t)} PC_t,$
      $\tilde{G}_{k(t)} \leftarrow \tilde{G}_{k(t)} - u_t^1(\Phi),\ \tilde{P}_{k(t)} \leftarrow \tilde{P}_{k(t)} - 1$;
      update solution as follows:
      $\tilde{y}_t = 1, \quad \tilde{v}_t \leftarrow \tilde{v}_t + u_t^1(\Phi), \quad \tilde{x} \leftarrow$ DEMAND_ASSIGNMENT($\mathcal{S}$, t);
      $CL \leftarrow \{j \in M_B | \tilde{y}_j = 0,\ \tilde{P}_{k(j)} > 0,\ \tilde{G}_{k(j)} \geq H_j\}$;
    end if
  end while
  $\mathcal{S} \leftarrow (\tilde{y}, \tilde{v}, \tilde{x})$;
return $(\mathcal{S})$
end procedure

3. In this procedure $\tilde{y}$ remains the same.

Pseudocode 6 Addition of new kernels

procedure OPEN_NEW_KERNELS($\mathcal{S}$, Φ)
  for $(k \in K | \tilde{G}_k > 0)$ **do**
    $CL \leftarrow \{j \in M_k | \tilde{V}_j > 0\}$;
    while $(\tilde{G}_k > 0$ and $CL \neq \emptyset)$ **do**
      compute $u_j^2(\Phi)$, PC_j and δ_j for each $j \in CL$ according to (28), (30), and (31);
      if $(\sum \delta_j \neq 0)$ **then**
        $t \leftarrow \arg\max_{j \in CL}\{\delta_j\}$;
        $C_t \leftarrow C_t + PC_t,\ \tilde{V}_t \leftarrow \tilde{V}_t - u_t^2(\Phi),\ CI_t \leftarrow CI_t + \beta_{k(t)} PC_t,\ \tilde{G}_{k(t)} \leftarrow \tilde{G}_{k(t)} - u_t^2(\Phi)$;
        update solution: $\tilde{v}_t \leftarrow \tilde{v}_t + u_t^2(\Phi)$, $\tilde{x} \leftarrow$ DEMAND_ASSIGNMENT($\mathcal{S}$, t);
        $CL \leftarrow \{j \in M_k | \tilde{V}_j > 0\}$;
      end if
    end while
  end for
  $\mathcal{S} \leftarrow (\tilde{y}, \tilde{v}, \tilde{x})$;
return $(\mathcal{S})$
end procedure

3.2.3 CM: Demand allocation

The DEMAND_ALLOCATION() procedure is shown in Pseudo-code 7. This procedure is required to allocate the demand points in a given solution. The procedure is divided into two main stages. In the first stage, demand is allocated to active sites with available capacity of the same institutions (Steps 2–13). Then, if there are still unallocated demand points, they are allocated to any active

Pseudocode 5 Assignment of demand

1: **procedure** DEMAND_ASSIGNMENT($\mathcal{S}$, t)
2:     **for** $k \in K$ **do**
3:         **if** $(k = k(t))$ **then**
4:             $DPL \leftarrow \{(k, i \in N) | d_{it} < \lambda_{ki}, \tilde{w}_{ki} \leq C_t\}$;
5:         **else**
6:             $DPL \leftarrow \{(k, i \in N) | d_{it} < \lambda_{ki}, \tilde{w}_{ki} \leq C_t, \tilde{x}_{k(t)i} = t, \tilde{w}_{ki} \leq CI_t, \tilde{w}_{ki} \leq O_k\}$;
7:         **end if**
8:     **end for**
9:     compute θ_{ki} for each $(k, i) \in DPL$ according to (32);
10:     **while** $(DPL \neq \emptyset)$ **do**
11:         $(r, l) \leftarrow \arg\max_{(k,i) \in DPL}\{\theta_{ki}\}$;
12:         update working parameters: $C_t \leftarrow C_t - \tilde{w}_{rl}$, $\tilde{w}_{rl} \leftarrow 0$, $\lambda_{rl} \leftarrow d_{lt}$;
            if $(r \neq k(t))$ **then**
                $CI_t \leftarrow CI_t - w_{rl}$, $O_r \leftarrow O_r - w_{rl}$;
            end if
13:         update solution as follows: $\tilde{x}_{rl} \leftarrow t$;
14:         update DPL according to Steps 2–8;
15:         compute θ_{ki} for each $(k, i) \in DPL$ according to (32);
16:     **end while**
17: **return** $(\tilde{x})$
18: **end procedure**

site, no matter the institution (Steps 14–30). In the first stage, the procedure is repeated for each institution, demand points with unallocated demand are added to the candidate list CL_1. Then, the demand point with the highest demand level is selected to be allocated to the nearest active site with enough capacity (Step 5). If there are no active sites with enough capacity of the same institution, the demand point is removed from CL_1 (Step 7), otherwise, the residual capacity and the solution are updated in Steps 9 and 10, respectively. The steps of the second stage are very similar, but in this case, demand points with unallocated demand of any institution are considered in the candidate list (CL_2). The demand point with the highest demand level is selected in Step 17 and then, the nearest active site that satisfies all the requirements is selected (Step 18–22). In this step, if the demand point and the active site belong to different institutions, additional requirements must be validated ($\tilde{x}_{k(j)l} = j$, $\tilde{w}_{rl} \leq CI_j$, $\tilde{w}_{rl} \leq O_r$). If there are no feasible active sites, the demand point is removed from CL_2 in Step 24; otherwise, the working parameters and the solution are updated to allocate this demand point (r, l) to the active site t in Steps 26 and 27, respectively. Then, the demand point is removed from CL_2 and the process is repeated until CL_2 becomes empty.

Pseudocode 7 Demand allocation

```
procedure DEMAND_ALLOCATION(\mathcal{S})
    for ( k ∈ K ) do
        CL_1 ← {i ∈ N | \tilde{w}_{ki} > 0};
        while (CL_1 ≠ ∅) do
            t ← 0; l ← arg max_{i∈CL}{\tilde{w}_{ki}}; t ← arg min_{j∈M_k}{d_{lj} | \tilde{w}_{kl} ≤ C_j};
            if (t = 0) then
                CL_1 ← CL_1 \ {l} ;
            else
                update working parameters: C_t ← C_t − \tilde{w}_{kl},   \tilde{w}_{kl} ← 0,   λ_{kl} ← d_{lt};
                update solution: \tilde{x}_{kl} ← t;
            end if
        end while
    end for
    CL_2 ← all pairs (k, i) from i ∈ N and k ∈ K such that: \tilde{w}_{ki} > 0;
    while (CL_2 ≠ ∅) do
        t ← 0;
        (r, l) ← arg max_{(k,i)∈CL_2}{\tilde{w}_{ki}};
        if (r = k(j)) then
            t ← arg min_{j∈M}{d_{lj} | \tilde{w}_{rl} ≤ C_j};
        else
            t ← arg min_{j∈M}{d_{lj} | \tilde{w}_{rl} ≤ C_j, \tilde{x}_{k(j)l} = j, \tilde{w}_{rl} ≤ CI_j, \tilde{w}_{rl} ≤ O_r};
        end if
        if (t = 0) then
            CL_2 ← CL_2 \ {(r, l)} ;
        else
            update working parameters: C_t ← C_t − \tilde{w}_{rl}, \tilde{w}_{rl} ← 0,
             if (r ≠ k(t)) then
                CI_t ← CI_t − w_{rl}, O_r ← O_r − w_{rl};
             end if
            update solution: \tilde{x}_{rl} ← j;
            CL_2 ← CL_2 \ {(r, l)} ;
        end if
    end while
return (\tilde{x})
end procedure
```

3.2.4 CM: Force the opening of new sites

In some partial solutions, for a given institution k, there is a special case when no more UCS can be selected. This special case occurs when all UCS require a greater number of kernels than the ones that are available in the institution ($\tilde{G}_k > H_j$). In this case, it is required to unassign kernels from some active sites and assign them to an UCS. This additional step is not required when CM is applied for the first time because the method prioritizes assigning kernels to CSs instead HCUs. However, when CM is applied over partial solutions that were randomly modified, this additional step may be sometimes needed. The procedure KERNELS_TRANSFER() is shown in Pseudo-code 8. For each institution such that $\tilde{P}_k > 0$, the candidate list CL_1 with UCSs is created (Step 4). In Step 6, the number of kernels to be transferred $(u_j^3(\Phi))$ and PC_j, and δ_j are computed. If there is no benefit $(\sum \delta_j = 0)$, another institution is selected to repeat the process. In the other case, the element with the highest benefit is selected (j_1) in Step 10. A second candidate list CL_2 is created to identify active sites such that some kernels could be released according to the criteria of Step 11. If CL_2 is empty, j_1 is removed from CL_1 and another element is chosen. Otherwise, the nearest element of CL_2 to j_1 is selected. The working parameters are updated in Step 17 and the solution is updated in Step 18. The sites with modified capacity (j_1 and j_2) are joined to a candidate list CL_3 in step 19. In Step 20, there is a subroutine that is required to deallocate demand of all the active sites that are near to j_1 and j_2. This subroutine is explained in the following paragraph. Then, the procedure shown in Pseudo-code 7 is called to allocate all the demand to complete a new feasible solution. All this process is repeated until $\sum_{k \in K} \tilde{P}_k = 0$ or there is no possibility to select more candidate sites.

Pseudocode 8 Kernels transfer

1: **procedure** KERNELS_TRANSFER($\mathcal{S}$, D_1)
2: select solution $\mathcal{S}$;
3: **for** ($k \in K | \tilde{P}_k > 0$) **do**
4: $CL_1 \leftarrow \{j \in M_{B_k} | \tilde{y}_j = 0\}$;
5: **while** ($\tilde{P}_k > 0$ and $CL_1 \neq \emptyset$) **do**
6: compute $u_j^3(\Phi)$, PC_j and δ_j for each $j \in CL_1$ according to (29), (30) and (31);
7: **if** ($\sum \delta_j = 0$) **then**
8: **break while**;
9: **end if**
10: $j_1 \leftarrow \arg\max_{j \in CL_1} \{\delta_j\}$;
11: $CL_2 \leftarrow \{j \in M_k | \tilde{v}_j \geq u_{j_1}^3$, **if** $(j \in M_{B_k})$ **then** $(\tilde{y}_j = 1, \tilde{v}_j - H_j \geq u_{j_1}^3)\}$;
12: **if** ($CL_2 = \{\emptyset\}$) **then**
13: $CL_1 \leftarrow CL_1 \backslash \{j_1\}$;
14: **go to** Step 10;
15: **end if**
16: $j_2 \leftarrow \arg\min_{j \in CL_2} \{d_{j_1 j_2}\}$;
17: update working parameters: $\tilde{V}_{j_1} \leftarrow \tilde{V}_{j_1} - u_{j_1}^3$, $\tilde{P}_k \leftarrow \tilde{P}_k - 1$, $\tilde{V}_{j_2} \leftarrow \tilde{V}_{j_2} + u_{j_1}^3$;
18: update solution: $\tilde{v}_{j_1} \leftarrow \tilde{v}_{j_1} + u_{j_1}^3$, $\tilde{y}_{j_1} \leftarrow 1$, $\tilde{v}_{j_2} \leftarrow \tilde{v}_{j_2} - u_{j_1}^3$;
19: $CL_3 \leftarrow \{j_1, j_2\}$;
20: $\tilde{x} \leftarrow$ DEMAND_DEALLOCATION($\mathcal{S}$, CL_3, D_1);
21: $\tilde{x} \leftarrow$ DEMAND_ALLOCATION($\mathcal{S}$);
22: $CL_1 \leftarrow CL_1 \backslash \{j_1\}$;
23: **end while**
24: **end for**
25: $\mathcal{S} \leftarrow (\tilde{y}, \tilde{v}, \tilde{x})$;
26: **return** ($\mathcal{S}$)
27: **end procedure**

Demand deallocation

Procedure DEMAND_DEALLOCATION() show in Pseudo-code 9 is required every time kernels are unassigned from active sites. The objective is to deallocate all demand points from active sites of CL_1, but also of other active sites that are nearby at a maximum distance D_1. In this case, D_1 is used to just modify the allocation decisions over an influence area instead deallocating the entire problem. This procedure is used as a step in Pseudo-code 8, in the deconstruction strategies, and in the local search procedures. In Step 2, all the involved active sites are stored in $CL2$. For each j of $CL2$, all demand point (k, i) such that $\tilde{x}_{ki} = j$ are deallocated, updating the associated working parameters in Steps 6 and 9, and the solution in Step 7. Then, the decision variable $\tilde{x}$ is returned.

Pseudocode 9 Deallocation of demand

1: **procedure** DEMAND_DEALLOCATION($\mathcal{S}$, CL_1, D_1)
2:     $CL_2 \leftarrow \{j \in M | d_{jt} \leq D_1$ for some $t \in CL_1\}$
3:     **for** $(j \in CL_2)$ **do**
4:         $DPL \leftarrow \{(k, i) | k \in K,\ i \in N,\ \tilde{x}_{ki} = j\}$;
5:         **for** $((k, i) \in DPL)$ **do**
6:             update working parameters:
                $\tilde{w}_{ki} \leftarrow w_{ki}$,
                if $(k \neq k(j))$ **then**
                    $O_k \leftarrow O_k + w_{ki}$;
                end if
7:             update solution as follows: $\tilde{x}_{ki} \leftarrow \infty$;
8:         **end for**
9:         update working parameters: $C_j \leftarrow KC(\tilde{v}_j)$, $CI_j \leftarrow \beta_{k(j)} C_j$;
10:     **end for**
11: **return** $(\tilde{x})$
12: **end procedure**

3.3 Deconstruction phase

Two deconstruction strategies are proposed for this problem: DS1 and DS2. The procedures and pseudo-codes are described in the following subsections.

3.3.1 Deconstruction strategy 1 (DS1)

In this first procedure, a random subset of SCSs is deselected in the solution and all the demand points in the influence area of these sites are deallocated. The first parameter to define is the number of sites to deselect in the solution according to ρ. The ceil of the multiplication between ρ and the total number of selected sites is used as a deconstruction parameter as is shown in the following equation:

$$n_1 = \lceil \sum_{j \in M_B} \tilde{y}_j \rho \rceil \tag{16}$$

The procedure is shown in Pseudo-code 10. The number of SCS to deselect is calculated with Equation 16. In Step 3, all candidate sites such that $\tilde{y}_j = 1$ are stored in CL_1. A random subset of n_1 elements is chosen in Step 4. Then, the associated working parameters are updated in Step 5 and the solution in Step 6. The subroutine DEMAND_DEALLOCATION() shown in Pseudo-code 9 is called for deallocating all the involved demand points. Finally, the initial solution has been partially destroyed and is returned as output.

Pseudocode 10 Deconstruction based on new sites

```
1:  procedure DS1(S, ρ, D_2)
2:      compute n_1 according to (16) and ρ;
3:      CL_1 ← {j ∈ M_B | ỹ_j = 1};
4:      CL_2 ← random(CL_1, n_1);
5:      for all j ∈ CL_2 update working parameters: G̃_k(j) ← G̃_k(j) + ṽ_j,  P̃_k(j) ← P̃_k(j) + 1, Ṽ_j ← V_j;
6:      for all j ∈ CL_2 update the solution as follows:  ỹ_j ← 0,  ṽ_j ← 0;
7:      x̃ ← DEMAND_DEALLOCATION(S, CL_2, D_2);
8:      S ← (ỹ, ṽ, x̃);
9:  return (S)
10: end procedure
```

3.3.2 Deconstruction strategy 2 (DS2)

In the second deconstruction strategy, for a given number of active sites, kernels are unassigned, and the related demand is also deallocated. For each $j \in M$, an auxiliary binary parameter (η_j) is used to determine if this site has assigned kernels. The equation is the following:

$$\eta_j = \begin{cases} 1 & \text{if } \tilde{v}_j > 0 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

The following equation determines the number of sites to unassing kernels:

$$n_2 = \lfloor \sum_{j \in M} \eta_j \rho \rfloor \tag{18}$$

The procedure of DS2 is shown in Pseudo-code 11. The number of active sites to remove kernels is computed in Step 2 and the list is stored in CL_1 in Step 3. A random subset of these elements is chosen in Step 4. For each element of CL_2, the working parameters associated with the capacity and the solution are updated (Steps 5–9), if some of them belong to M_B, there is a minimum number of kernels (H_j) that can not be removed to satisfy constraints (9). DEMAND_DEALLOCATION is called in Step 10 to deallocating demand of the involved sites, and the solution is returned.

Pseudocode 11 Deconstruction based on new kernels

procedure DS2($\mathcal{S}$, ρ, D_2)
  compute n_2 according to (18) and ρ;
  $CL_1 \leftarrow \{j \in M | \tilde{v}_j > 0\}$;
  $CL_2 \leftarrow \text{random}(CL1, n_2)$;
  for $(j \in CL_2)$ **do**
    compute u as follows: **if** $(j \in M_B)$ **then** $(u \leftarrow \tilde{v}_j - H_j)$ **else** $(u \leftarrow \tilde{v}_j)$;
    update working parameters: $\tilde{V}_j \leftarrow \tilde{V}_j + u, \quad \tilde{G}_{k(j)} \leftarrow \tilde{G}_{k(j)} + u$;
    update solution as follows: $\tilde{v}_j \leftarrow \tilde{v}_j - u$;
  end for
  $\tilde{x} \leftarrow$ DEMAND_DEALLOCATION($\mathcal{S}$, CL_2, D_2);
  $\mathcal{S} \leftarrow (\tilde{y}, \tilde{v}, \tilde{x})$;
return $(\mathcal{S})$
end procedure

3.4 Local search methods

Two neighborhoods are proposed for this problem. Each can be used as a stand-alone strategy, or within a VND scheme as described below.

Neighborhood 1

The move $m_1(j_1, j_2)$ is defined as transferring all the kernels of a candidate site $j_1 \in M_{B_k}$ such that $\tilde{y}_{j_1} = 1$ to another candidate site $j_2 \in M_{B_k}$ such that $\tilde{y}_{j_2} = 0$, $\tilde{v}_{j_1} \geq H_{j_2}$, and $\tilde{v}_{j_1} \leq V_{j_2}$. The neighborhood is the set of neighbors reachable from the solution $\mathcal{S}$ by performing all possible moves $m_1(j_1, j_2)$ for all $j_1 \in M_{B_k}$ such that $\tilde{y}_{j_1} = 1$. We propose for large instances to bound the neighborhood by considering only the R nearest sites from j_1.

Neighborhood 2

The move $m_2(j_1, j_2)$ is defined as transferring the largest amount of kernels from a HCU $j_1 \in M_A$ such that $\tilde{v}_{j_1} > 0$ to another HCU $j_2 \in M_{A_{k(j_1)}}$ such that $\tilde{V}_{j_2} > 0$. The neighborhood is the set of neighbors reachable from the solution $\mathcal{S}$ by performing all possible moves $m_2(j_1, j_2)$ for all $j_1 \in M_A$ such that $\tilde{v}_{j_1} > 0$. We also propose for large instances to bound the neighborhood by considering only the R nearest sites from j_1.

Local search 1

The LS1 procedure is shown in Pseudo-code 13 in the Appendix. In this procedure, SCSs are considered to be unassigned from the solution. For each $j \in M_B | \tilde{y}_j = 1$, a list of UCS sites is created (CL_2). This list is composed of the R nearest sites to j_1 of the same institution. Then, the kernels are transferred to this site, and the allocation of demand must be adjusted. To this end, the DEMAND_DEALLOCATION() and DEMAND_ALLOCATION() procedures are called to solve again the allocation subproblem for the involved demand points. These steps generate a feasible solution that is compared with the best solution found so far. If the objective value of the current solution is better, the best solution is updated; else, the new solution is discarded and another element of CL_2 is evaluated. If any site of $CL2$ produces a better solution, the element j_1 of CL_1 is removed and another one is evaluated. The procedure ends, when all the elements of CL_1 were evaluated and there are no more interchanges that produce an improvement in the objective function. There is also a time limit that can be used if the local optima consume a significant amount of computing time.

Local search 2

The complete procedure for LS2 is shown in Pseudo-code 14 in the Appendix. In this local search, only HCUs are considered for transferring kernels to other HCUs of the same institution. In preliminary experiments, we found that including candidate sites does not generate a significant improvement. The steps are very similar to LS1 with slight differences. A first list CL_1 composed of all the HCUs with assigned kernels is created. Then, one element of this list is selected (j_1). A second list (CL_2) of the R nearest HCUs to j_1 of the same institution is created. Then, the kernels are transferred and all the involved demand in the area is deallocated and the allocation procedure is called to complete a feasible solution. If the interchange produces a better solution, the best solution is updated. The procedure ends when neither interchange produces an improvement or when a computing time limit is reached.

Variable Neighborhood Descent

In the IG, the LS procedure is typically applied at each iteration. Though, a most robust method such as a VND can be applied. The VND is a strategy of the variable neighborhood search procedure where the local searches are performed in a systematic way. Different neighborhoods are explored sequentially. Typically, one explores first the least expensive to evaluate and so on. The process iterates over each neighborhood while improvements are found, applying the local search until meeting a local optima at each neighborhood. Then, the final solution is a local optima of all the explored neighborhoods. However, we are dealing with large instances, and finding a local optimal may consume a lot of resources. Therefore, a time limit is defined at each local search procedure to reduce the computing time leaving off to find the local optima in some cases. Pseudocode 12 shows the VND procedure.

Pseudocode 12 Variable Neighborhood Descent

procedure VND($\mathcal{S}$)
  $\mathcal{S}^* \leftarrow \mathcal{S}$
  $t \leftarrow 1$;
  while $(t \leq t^{\max})$ **do**
    $\mathcal{S} \leftarrow \text{LS\_}t(R, time\_limit)$;
    if $(Z(\mathcal{S}) < Z(\mathcal{S}^*))$ **then**
      $\mathcal{S}^* \leftarrow \mathcal{S}$;
      $t \leftarrow 1$;
    else
      $t \leftarrow t + 1$;
    end if
  end while
return $(\mathcal{S}^*)$
end procedure

3.5 Optimization of the allocation subproblem (ALLOP)

Note that when fixing the capacity decision variables as $y_j = \tilde{y}_j \; \forall j \in M_B$ and $v_j = \tilde{v}_j \; \forall j \in M$, we are left with an allocation subproblem (ALLOP) that is easier to solve because it is considerable smaller. Not only many integer variables are eliminated from this model but many constraints become redundant as well. For instance, Constraints (7)-(12), (14), and (15) of MIFLUP are not considered in this subproblem because they are already satisfied. The resulting linear binary model has a single decision variable type x_{kij}. In this supbroblem, the right-hand side of constraints (22) and (23) is constant. The set M is also reduced to the subset $M^* = \{j \in M | a_j + \tilde{v}_j > 0\}$.

We suggest solving this problem with an exact method as a final step of a heuristic solution of MIFLUP. The solution to this subproblem will optimize the allocation of demand to HCUs. Furthermore, since we have an entirely feasible solution from MIFLUP, the heuristic values $\tilde{x}_{kij}$ can be provided as input to an exact method.

The proposed ALLOP subproblem is given by:

$$\text{Minimize} \quad f(x) = \sum_{i \in N} \sum_{j \in M^*} \sum_{k \in K} w_{ki} d_{ij} x_{kij} \tag{19}$$

$$\text{subject to:} \quad \sum_{j \in M^*} x_{kij} = 1 \qquad k \in K, \; i \in N \tag{20}$$

$$\sum_{r \in K : r \neq k(j)} x_{rij} \leq (|K| - 1) x_{kij} \qquad k \in K, \; i \in N, \; j \in M_k^* \tag{21}$$

$$\sum_{i \in N} \sum_{k \in K} w_{ki} x_{kij} \leq KC(a_j + \tilde{v}_j) \qquad j \in M^* \tag{22}$$

$$\sum_{i \in N} \sum_{r \in K : r \neq k(j)} w_{ri} x_{rij} \leq KC(a_j + \tilde{v}_j) \beta_{k(j)} \qquad j \in M^* \tag{23}$$

$$\sum_{i \in N} \sum_{j \in M_k^*} w_{ki} x_{kij} \geq \gamma_k \sum_{i \in N} \sum_{k \in K} w_{ki} \qquad k \in K \tag{24}$$

$$x_{kij} \in \{0, 1\} \qquad k \in K,\ i \in N,\ j \in M^* \tag{25}$$

4 Empirical assessment

The proposed metaheuristic was evaluated using the data sets provided by Mendoza-Gómez and Ríos-Mercado [24]. This data is based on a case study applied in the State of Mexico, Mexico. This state is composed of 125 counties that are grouped into 19 jurisdictions as shown in Figure 4. Four health care institutions are evaluated, SSA and IMSS-Bienestar (I1) were considered as a single institution for the uninsured population, IMSS (I2) for private sector workers, ISSSTE (I3) for federal workers, and ISSEMyM (I4) for state-level workers. Table 2 shows a summary of the number of demand points, HCUs, and candidate sites by jurisdiction. There are in total more than eight thousand demand points and a little over a thousand and four hundred HCUs in the state. Candidate sites were selected in places where there are no HCUs and with a minimum population size of over five hundred inhabitants. The distribution of demand points, HCUs, and candidate sites can be seen in Figure 5. I1 has the higher number of HCUs in the state with 1200 HCUs, I2 and I4 are nearly one hundred HCUs, and I3 has only 37 HCUs. In Table 3, the actual capacity is compared with the demand assuming that a basic kernel can serve up to 3,000 inhabitants. As can be seen in the actual demand covered, the first three institutions have lower capacity than demand and I4 has 165% of additional capacity regarding the demand to be covered. In the problem, new basic kernels can be added to the system to increase the system's capacity and reduce the total travel distance. In the experiments, new kernels can be added to each institution by jurisdiction to allocate all demand points and to avoid infeasible solutions. In the table, we show the additional basic kernels that we are suggesting to open for each institution. This additional capacity allows to have enough capacity for institution to cover internal demand as it can be seen in the last row. However, inter-institutional allocation can help to improve the access in regions where there is not enough capacity for a given institution.

The proposed solution methods are tested in 18 instances created by grouping adjacent jurisdictions. The main characteristics of these instances are shown in Table 4. The range of demand points (DP) is between 547 to 2,940, the HCUs range is between 71 to 565, and the candidate sites range is between 103 to 461. For the results of the following subsections, we use the average values of this instance set.

All procedures were coded in C++ and compiled with Visual Studio 2019, and run on a PC with 2.30 GHz Intel Core i7-4712HQ processor, and 16GB of RAM. A C++ application with Concert Technology of ILOG CPLEX 20.1.0 was used to call the B&B algorithm.

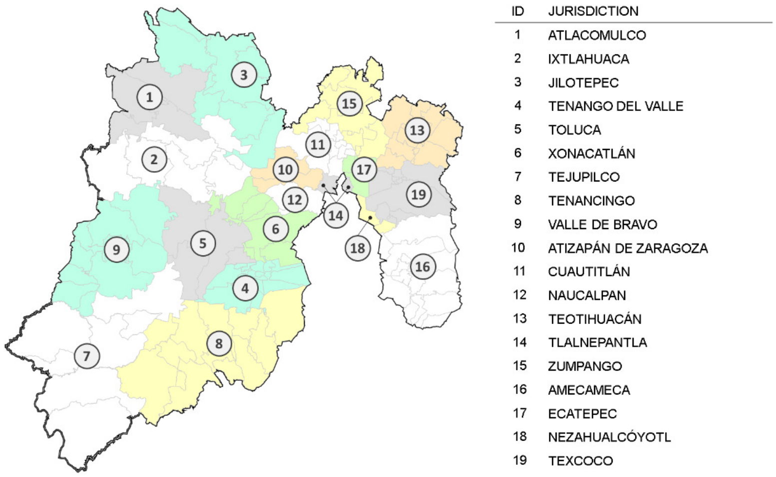

Figure 4: Identification of jurisdiction in the State of Mexico.

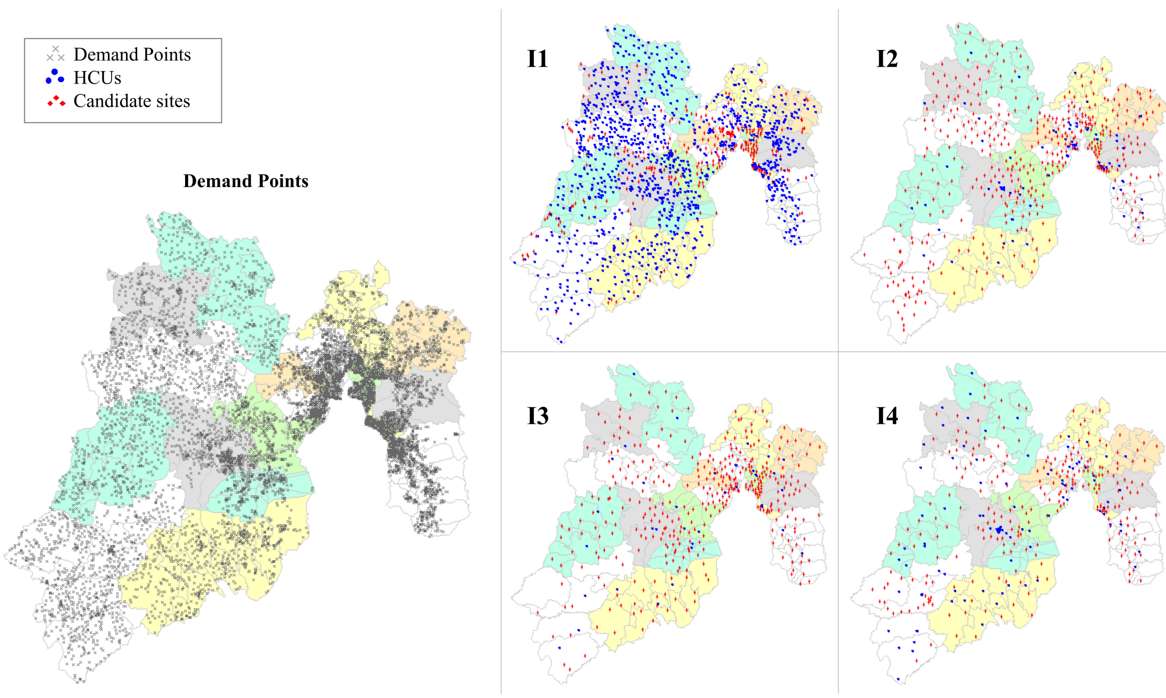

Figure 5: Location of demand points, HCUs, and candidates sites in the State of Mexico.

4.1 Fine-tuning of constructive method

The goal of this experiment is to fine tune the algorithmic parameters for the constructive method. All instances were tested with the constructive method evaluating the three strategies $\Phi =$(i), (ii), and (iii), and different distance bounds $D_1 = 5$, 10, 15, 20, and 30 kilometers that represents the influence area for sites whose capacity is modified. The solution was initialized with $D_0 = 15$ kilometers. Figure 6 shows the performance of the constructive method comparing the objective

Table 2: Instances size by jurisdiction.

Jurisdiction	Demand Points	HCUs					Candidate Sites				
		I1	I2	I3	I4	Total	I1	I2	I3	I4	Total
1	327	80	2	1	5	88	11	23	13	10	57
2	523	157	2	2	5	166	23	37	21	16	97
3	331	76	2	2	5	85	4	23	13	10	50
4	269	50	3	2	2	57	5	16	19	8	48
5	466	88	9	2	17	116	29	19	29	14	91
6	395	81	4	3	4	92	20	28	16	12	76
7	802	74	2	3	12	91	8	43	14	26	91
8	584	97	5	3	10	115	11	23	26	18	78
9	424	82	3	2	7	94	22	17	25	13	77
10	272	30	3	1	5	39	11	16	19	8	54
11	574	38	6	2	5	51	33	31	23	17	104
12	275	24	5	1	2	32	13	11	17	8	49
13	341	46	1	2	1	50	7	22	14	10	53
14	205	22	8	1	1	32	12	8	14	6	40
15	418	50	2	2	3	57	6	20	20	13	59
16	789	78	10	2	5	95	7	17	26	24	74
17	468	33	9	3	3	48	33	19	28	14	94
18	265	30	5	2	3	40	19	16	11	8	54
19	425	64	6	1	2	73	9	17	23	13	62
Total	8,153	1,200	87	37	97	1,421	283	406	371	248	1,308

Table 3: Actual and potential capacity in the system.

	I1	I2	I2	I4	Total
Demand (x1000)	9,652	4,467	717	306	15,143
Actual basic kernels	2,448	1,194	170	270	4,082
Potential basic kernels	1,130	438	136	3	1,707
Actual demand covered (%)	76	80	71	265	81
Potential demand covered (%)	111	110	128	268	115

Table 4: Instances size by jurisdiction.

n	Jurisdictions	DP	HCU					Candidate Sites				
			I1	I2	I3	I4	Total	I1	I2	I3	I4	Total
1	1,2	850	237	4	3	10	254	34	60	34	26	154
2	3,11	905	114	8	4	10	136	37	54	36	27	154
3	4,8	853	147	8	5	12	172	16	39	45	26	126
4	5,6	861	169	13	5	21	208	49	47	45	26	167
5	7,9	1,226	156	5	5	19	185	30	60	39	39	168
6	10,12	547	54	8	2	7	71	24	27	36	16	103
7	14,17	673	55	17	4	4	80	45	27	42	20	134
8	13,15	759	96	3	4	4	107	13	42	34	23	112
9	16,18,19	1,479	172	21	5	10	208	35	50	60	45	190
10	1,2,3	1,181	313	6	5	15	339	38	83	47	36	204
11	7,8	1,386	171	7	6	22	206	19	66	40	44	169
12	4,5,9	1,159	220	15	6	26	267	56	52	73	35	216
13	6,10,12	942	135	12	5	11	163	44	55	52	28	179
14	11,14,15,17	1,665	143	25	8	12	188	84	78	85	50	297
15	13,16,18,19	1,820	218	22	7	11	258	42	72	74	55	243
16	1,2,3,10,11,12	2,302	405	20	9	27	461	95	141	106	69	411
17	4,...,9	2,940	472	26	15	52	565	95	146	129	91	461
18	13,...,19	2,911	323	41	13	18	395	93	119	136	88	436

function value and the computing time for each set of instances combining different values of Φ and D_1. In the left-hand side plot, we can observe that the best performance was obtained with a distance bound of $D_1 = 15$ kilometers. The best objective values and the shortest interquartile ranges were obtained with this bound. Although strategy $\Phi =$(iii) produced the lowest objective values, the difference is not significant. The computing time increases as D_1 increases, as it is observed in the right-hand side plot. For $D_1 = 15$ km, the computing time varies between 0.2 and 3.8 seconds showing a very small difference in favor with $\Phi =$(i) with an average time of 1.003 seconds. Therefore, for the following experiments $\Phi =$(i) and $D_1 = 15$ km are used.

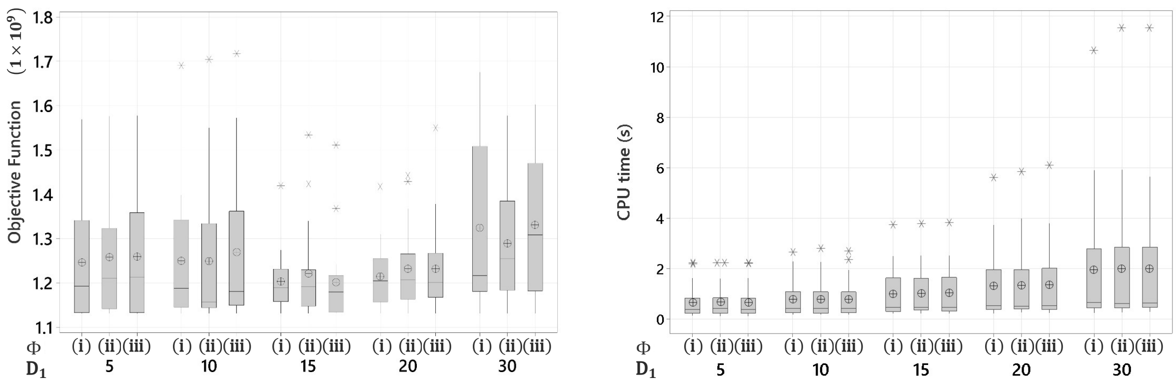

Figure 6: Result of the constructive method.

4.2 Iterated Greedy Algorithm with Deconstruction Strategies

The objective of these experiments is to fine-tune the percentage of deconstruction, the number of iterations for the IG, and the algorithmic parameter D_2 for the deconstruction procedures. To this end, the deconstruction strategies DS1 and DS2 were evaluated with the proposed methods H1 and H2 in Table 1. The ρ values were 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. The number of iterations was fixed to 100 and the values of D_2 for deconstruction procedure were 2.5, 5, 7.5, and 10 km. Table 5 shows the average relative improvements of the tested instances with respect to the initial solution found by the constructive method with $\Phi =$(i), $D_0 = 15$ km, and $D_1 = 15$ km. As can be seen, the best results are found with $\rho = 0.2$ and $D_2 = 2.5$ km with an average improvement of 6.9% and 14.3% using the methods H1 and H2, respectively. The best results were found with the method H2. This can be attributed to the fact that H2 modifies the kernel assignment of HCUs and selected candidate sites, while the method H1 only takes into account candidate sites. Figure 7 shows a boxplot of the iterations required to find the best solution for H1 and H2 for each value of ρ. For $\rho = 0.20$, the maximum number of iterations was lower than 40 and 20 for H1 and H2, respectively. Therefore, in the following experiment, we consider that 50 iterations of the IG are enough to get good results for the tested instances.

4.3 Local Search

In these experiments, the two neighborhoods are evaluated within a simple local search algorithm. For both heuristics, the fine-tuned algorithmic parameters are R, $time\_limit$, and D_1. For the

Table 5: Assessment of H1 and H2 in terms of relative improvement.

Method	D_2	ρ 0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	Global
H1 IG{DS1, CM,–}	2.5	4.5	5.6	6.9	6.1	5.5	5.2	4.9	4.4	3.7	2.7	1.6	4.6
	5.0	4.0	4.8	6.2	5.1	4.5	4.7	3.4	2.7	2.3	1.7	1.3	3.7
	7.5	4.5	5.5	6.0	4.8	5.1	3.8	3.3	2.8	2.4	1.6	1.5	3.7
	10.0	4.8	5.7	6.3	5.6	4.7	4.4	3.6	2.8	2.5	1.8	1.2	3.9
	Global	4.4	5.4	6.3	5.4	5.0	4.5	3.8	3.2	2.7	1.9	1.4	4.0
H2 IG{DS2, CM,–}	2.5	14.0	13.8	14.3	11.4	11.1	9.4	8.2	6.5	5.5	3.2	1.7	9.0
	5.0	13.2	13.3	11.2	8.9	7.2	6.6	4.9	3.9	3.2	2.9	2.3	7.1
	7.5	13.0	10.6	8.4	7.1	5.4	4.9	3.9	3.8	3.2	2.6	2.3	5.9
	10.0	11.7	10.2	8.5	5.4	5.1	4.1	3.5	3.0	3.4	2.5	2.5	5.4
	Global	7.5	8.4	10.4	6.1	4.7	4.6	5.5	3.2	2.3	3.7	1.5	5.4
		13.0	12.0	10.6	8.2	7.2	6.3	5.1	4.3	3.8	2.8	2.2	6.9

Table 6: Assessment of H1 and H2 in terms of running time (CPU seconds).

Method	D_2	ρ 0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	Global
H1 IG{DS1, CM,–}	2.5	24	27	32	35	39	42	45	49	53	55	60	42
	5.0	27	32	38	42	47	51	55	59	62	65	70	50
	7.5	30	35	41	46	52	56	59	63	67	70	74	54
	10.0	33	37	44	48	53	59	61	64	69	72	76	56
	Global	29	33	39	43	48	52	55	59	63	66	70	50
H2 IG{DS2, CM,–}	2.5	26	28	32	35	37	39	40	42	43	43	43	37
	5.0	29	33	37	39	41	43	44	45	45	46	46	41
	7.5	32	36	40	42	43	45	45	46	46	46	47	42
	10.0	34	38	41	44	45	46	47	49	50	51	52	45
	Global	30	34	37	40	41	43	44	45	46	47	47	41

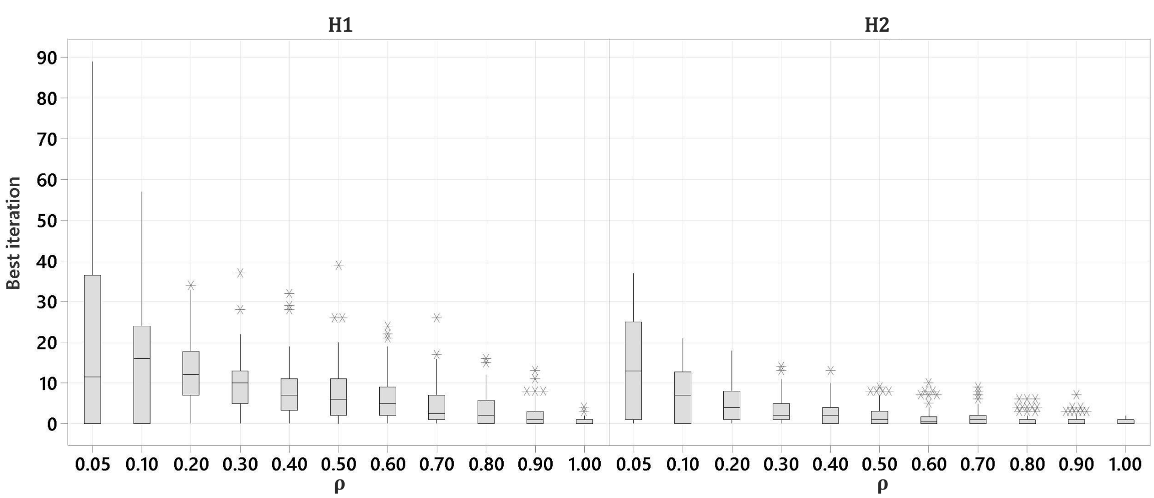

Figure 7: Assessment of H1 and H2 in terms of number of iterations required to find best solution.

first parameter, values of 1, 3, 5, and 10 sites were considered. Two contrasting time limits were evaluated: 60 seconds and 1 hour. The first time limit is proposed thinking about the algorithm being implemented in the iterative process, and the second time limit is proposed to evaluate the potential of the local search as a simple solution method. In these experiments, the initial solution was obtained with CM with Φ =(i), $D_0 = 15$ km, and $D_1 = 15$ km. Table 7 shows the average percentage of improvement and the average computing time for each value of R and each time limit. In all the cases, LS2 has better performance in the solution improvement and computing time. For LS1 the cost related to the computing time is very high since 1.70% of additional improvement is reached when the time limit is changed from 60 seconds to 1 hour, with a difference in the average computing time of 472 seconds. In the case of LS2, the additional improvement is about 2.1%, but the average time changed from 17 seconds to only 119 seconds, despite the time limit being set to 3,600 seconds. This means that local optima were found in most of the cases. The best improvement for LS1 was found with $R = 5$ for a time limit of 60 seconds, and $R = 10$ for a time limit of one hour. For LS2, the best results were found with $R = 10$ in both cases. In the following experiment for LS1 and LS2, we fixed the input parameters to $R = 5$ and $R = 10$, respectively.

Table 7: Assessment of local search.

		Average improvement (%)				Average CPU time (s)			
		R				R			
	Time limit	1	3	5	10	1	3	5	10
LS1	60 s	1.9	4.1	4.2	4.0	14	36	43	46
	1 h	1.9	5.0	6.3	7.7	43	281	718	994
LS2	60 s	2.0	4.6	5.2	7.9	3	17	20	28
	1 h	2.0	4.7	7.6	13.6	3	22	65	389

4.4 Assessment of the Iterated Greedy Algorithm

In this experiments, the propose heuristic methods shown in Table 1 are applied to the instance set with the fine-tuned parameters of previous experiment. For the ALLOP optimization, a MIP start strategy is used, this means that the heuristic solution of variables $\tilde{x}_{ki}$ is used as the initial solution of $x_{kij}$for the optimization, reducing the computing time. All the experiments were run with 50 iterations and a computing time limit of one-hour for the IG and one hour for the solution of ALLOP (using CPLEX). The solution's performance comparing the average improvement (%) (taking as reference constructive method solution with Φ =(i), $D_0 = 15$ km, and $D_1 = 15$ km) and the average computing times are shown in Figure 8 for each method. Better results are found when DS1 and DS2 are applied in that order into the IG with an average improvement of 14.3%. The best performance when the local search is integrated in the IG is found using the VND12 with an average improvement of 24.3% requiring 3,024 seconds on average. The allocation optimization was applied to these two last methods in H9 and H10. Similar results were found using VND12 and VND12; with an average improvement of 43.2% and 5,834 seconds on average for H9, and an average improvement of 42.9% in 5,835 seconds on average for H10.

To evaluate the algorithmic components, some experiments were carried out to show the contribution of each component in metaheuristic H9 which offers the best improvement. Table 8 shows the objective function values of different heuristic methods present in H9. The most basic heuristic

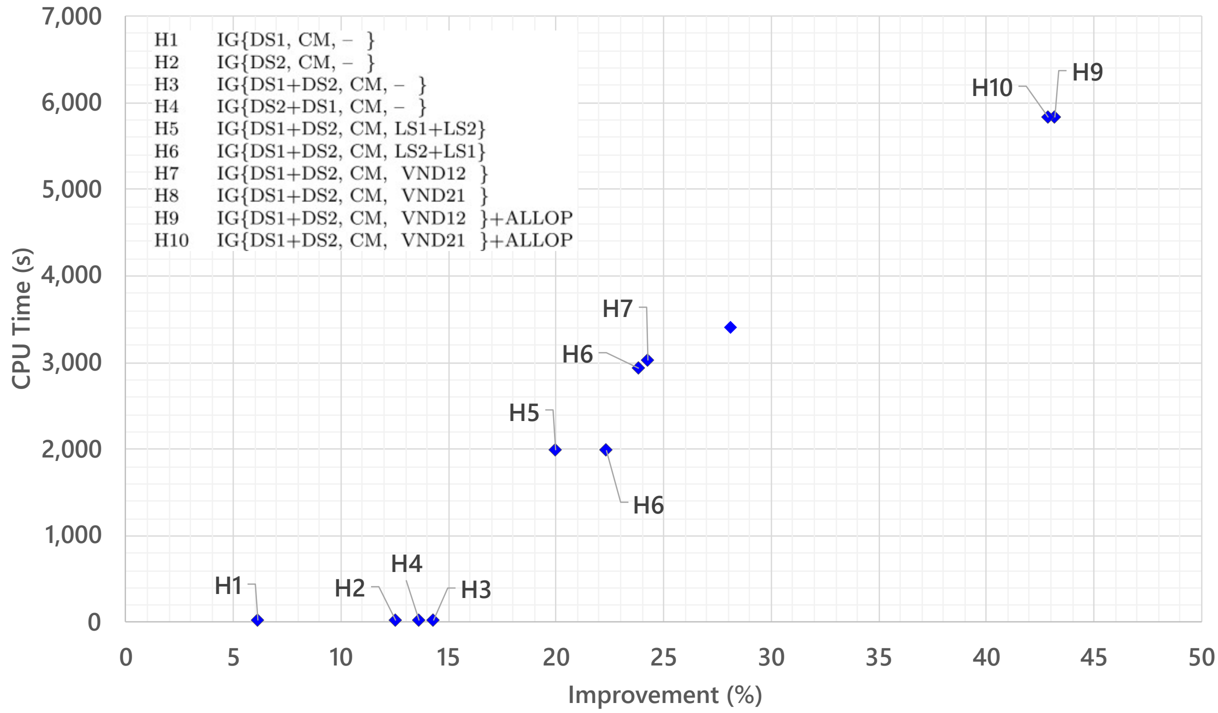

Figure 8: Average results applying different strategies.

is the CM in column 2, in H2, the CM was embedded in the IG, H7 includes the VND in the IG, and H9 represents the final metaheuristic that includes the ALLOP optimization. The IG reduces the solution of CM by 24.3% on average. The addition of VND12 decreases on average 12.0% the solutions obtained with the IG. The ALLOP optimization reduces 23.9% on average the solutions of the heuristic $H7$. In general, the most complex heuristic ($H9$) decreases by 43.2% on average the objective function values from the simple constructive method. The Friedman test, a non-parametric statistical test to identify differences in treatments across multiple test attempts, was applied to these experiment. The results fo this test showed that there was a significant difference in the median of the objective function values between the solutions of the heuristic methods using a 95% confidence level. Concluding with this, each component in the metaheuristic contributes to the improvement of the solutions. In the sixth column, the objective function values (OFV) obtained with the exact method under a 2-hours time limit are also provided to compare the results. NFS indicates a no feasible solution found with the exact method. The relative gap is shown in the last column as a reference of solution quality. Comparing the solutions of the exact method with the hybrid metaheuristic $H9$, only 6 out of 11 solved instances were better using the exact method with a very slight difference. However, the exact method was not able to find solutions in 7 out of 18 instances tested, most of them the largest instances.

Table 8: Assessment of different algorithmic components (solutions 1×10^6 km).

Instances	CM	(H2) IG{DS1+DS2,CM,–}	(H7) IG{DS1+DS2,CM,VND12}	(H9) (3)+ALLOP(1h)	B&B(2h) OFV	B&B(2h) Relative Gap
1	1.38	1.23	1.16	0.85	0.83	4.58
2	6.17	5.58	4.72	3.46	28.92	89.78
3	1.68	1.58	1.43	0.94	0.96	10.36
4	10.71	8.70	6.99	4.21	NFS	–
5	1.14	1.14	1.13	0.76	0.75	0.05
6	6.62	5.79	5.34	4.11	3.82	1.15
7	13.17	12.29	12.03	10.57	10.55	2.60
8	8.50	6.29	4.91	3.46	3.13	28.34
9	16.16	11.65	8.43	7.57	NFS	–
10	2.01	1.80	1.71	1.18	3.43	68.46
11	1.73	1.60	1.60	1.06	1.04	1.58
12	6.76	6.14	4.98	3.32	14.34	100.00
13	11.59	10.92	10.32	6.34	29.42	100.00
14	30.21	21.29	18.45	17.68	NFS	–
15	19.51	13.83	9.82	8.66	NFS	–
16	15.27	13.04	11.47	10.15	NFS	–
17	11.83	10.73	9.18	7.32	NFS	–
18	43.56	30.89	28.23	25.44	NFF	–

5 Conclusions

In this paper, we proposed an algorithmic framework for solving the multi-institutional regionalization of the primary HCUs problem. With this framework, several components were developed and assessed. The best results were found by an iterated greedy algorithm with a variable neighborhood descent procedure enhanced by an exact optimization of the allocation sub-problem. This allocation subproblem is obtained by fixing some location decisions beforehand. This metaheuristic solved instances of up to 3,000 demand points that are difficult to solve with exact methods such as the B&B algorithm of commercial solvers. The IG is based on a constructive method that systematically selects new sites, adds new kernels to the systems, and solves the demand allocation heuristically. This method generates a greedy feasible solution. Two deconstructive strategies are proposed: the first one randomly removes new sites of the solution and the second one removes assigned kernels of random sites. For the VND, two neighborhoods are proposed. The first one is based on the interchange of selected and unselected candidate sites, and the other one is based on the interchange of kernels between a pair of sites. Some mechanisms are considered to reduce the computing time of the complete method since it is proposed for large-scale instances. Although the complete metaheuristic produced the best results, alternative variations are also evaluated.

After fine tuning its individual components, the metaheuristic was applied to a case study based on the State of Mexico public health care system. With this information, eight instances with a range between 547 to 2,940 demand points were solved. For the IG, 20% of deconstruction and a maximum of 50 iterations provide good performance in both deconstrutive strategies. A time limit was set up for the IG iterations and another hour was set for the allocation optimization. The complete metaheuristic generated an improvement of 43.2% on average in the quality of solutions regarding the CM solutions. The simple IG produced an average improvement of 14.6% regarding the CM solutions. The inclusion of the VND strategy generated an additional improvement of 9.7% on average, and the allocation optimization generated an additional improvement of 18.8% on average. The instances were also compared with an exact algorithm, CPLEX branch-and-bound method under a running time limit of two hours. No optimal solutions were found and solutions were only found for 11 out of 18 instances. In 6 out of 11 instances, better solutions were found with CPLEX with an average improvement of 3.9% with respect to the metaheuristic, although at a much higher computing time. In the other instances, the metaheuristic provided better solutions. While the algorithmic parameters have been fine-tuned for this specific case study, clearly, further experiments are necessary for implementing the metaheuristic in different classes of instances.

For future research, alternative metaheuristics that take advantage of the proposed constructive method can be also implemented and assessed. The importance of providing good quality solutions is the direct effect on the access and quality of these types of services. Additionally, the developed metaheuristic can be used for other problems that share characteristics of the addressed problem such as the capacitated location-allocation features in a segmented system.

Acknowledgments: The research of the first author was supported by a postdoctoral fellowship from the Mexican Council of Humanities, Sciences and Technologies (CONAHCyT), and by Tecnológico de Monterrey. The second author was supported by UANL (grants UANL-PAICYT CE1416–20 and CE1837-21) and CONAHCyT (grants FC-2016-2/1948 and CF-2023-I-880).

References

[1] S. Ahmadi and I.H. Osman. Greedy random adaptive memory programming search for the capacitated clustering problem. *European Journal of Operational Research*, 162(1):30–44, 2005.

[2] A. Ahmadi-Javid and N. Ramshe. A stochastic location model for designing primary healthcare networks integrated with workforce cross-training. *Operations Research for Health Care*, 24: 100226, 2020.

[3] A. Ahmadi-Javid, P. Seyedi, and S.S. Syam. A survey of healthcare facility location. *Computers & Operations Research*, 79:223–263, 2017.

[4] R. Baldacci, E. Hadjiconstantinou, V. Maniezzo, and A. de Mingozzi. A new method for solving capacitated location problems based on a set partitioning approach. *Computers & Operations Research*, 29(4):365–386, 2002.

[5] M. Boccia, A. Sforza, C. Sterle, and I. Vasilyev. A cut and branch approach for the capacitated p-median problem based on fenchel cutting planes. *Journal of Mathematical Modelling and Algorithms*, 7(1):43–58, 2008.

[6] A. Casado, S. Bermudo, A.D. López-Sánchez, and J. Sánchez-Oro. An iterated greedy algorithm for finding the minimum dominating set in graphs. *Mathematics and Computers in Simulation*, 207:41–58, 2023.

[7] A. Ceselli and G. Righini. A branch-and-price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, 2005.

[8] A.R. de Aguiar, V.M. Fo, and L. da Silva Mota. Optimization models in the location of healthcare facilities: A real case in Brazil. *Journal of Applied Operational Research*, 4(1): 37–50, 2012.

[9] J.A. Díaz and E. Fernández. Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research*, 169(2):570–585, 2006.

[10] X. Feng, F. Zhao, G. Jiang, T. Tao, and X. Mei. A tabu memory based iterated greedy algorithm for the distributed heterogeneous permutation flowshop scheduling problem with the total tardiness criterion. *Expert Systems with Applications*, 238:121790, 2024.

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

[12] K. Fleszar and K. S. Hindi. An effective VNS for the capacitated p-median problem. *European Journal of Operational Research*, 191(3):612–622, 2008.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[14] M. Gnägi and P. Baumann. A matheuristic for large-scale capacitated clustering. *Computers & Operations Research*, 132:105304, 2021.

[15] O. Gokalp. An iterated greedy algorithm for the obnoxious p-median problem. *Engineering Applications of Artificial Intelligence*, 92:103674, 2020.

[16] P.M. Griffin, C.R. Scherrer, and J.L. Swann. Optimization of community health center locations and service offerings with statistical need estimation. *IIE Transactions*, 40(9):880–892, 2008.

[17] W. Gu, X. Wang, and S.E. McGregor. Optimization of preventive health care facility locations. *International Journal of Health Geographics*, 9(1):17, 2010.

[18] J. Hoffmann, J.S. Neufeld, and U. Buscher. Iterated greedy algorithms for customer order scheduling with dedicated machines. *IFAC-PapersOnLine*, 55(10):1594–1599, 2022.

[19] G. Laporte, S. Nickel, and F. Saldanha da Gama, editors. *Location Science*. Springer, Cham, Switzerland, 2nd edition, 2019.

[20] F. Liu, G. Li, C. Lu, L. Yin, and J. Zhou. A tri-individual iterated greedy algorithm for the distributed hybrid flow shop with blocking. *Expert Systems with Applications*, 237:121667, 2024.

[21] V. Maniezzo, A. Mingozzi, and R. Baldacci. A bionomic approach to the capacitated p-median problem. *Journal of Heuristics*, 4(3):263–280, 1998.

[22] V. Marianov and P. Taborga. Optimal location of public health centres which provide free and paid services. *Journal of the Operational Research Society*, 52(4):391–400, 2001.

[23] V. Marianov, M. Ríos, and P. Taborga. Finding locations for public service centres that compete with private centres: Effects of congestion. *Papers in Regional Science*, 83(4):631–648, 2004.

[24] R. Mendoza-Gómez and R.Z. Ríos-Mercado. Regionalization of primary health care units with multi-institutional collaboration. *Socio-Economic Planning Sciences*, 83:101343, 2022.

[25] R. Mendoza-Gómez and R.Z. Ríos-Mercado. Location of primary health care centers for demand coverage of complementary services. *Computers & Industrial Engineering*, 169:108237, 2022.

[26] R. Mendoza-Gómez, R.Z. Ríos-Mercado, and K.B. Valenzuela-Ocaña. An efficient decision-making approach for the planning of diagnostic services in a segmented healthcare system. *International Journal of Information Technology & Decision Making*, 18(5):1631–1665, 2019.

[27] R. Mendoza-Gómez, R.Z. Ríos-Mercado, and K.B. Valenzuela-Ocaña. An iterated greedy algorithm with variable neighborhood descent for the planning of specialized diagnostic services in a segmented healthcare system. *Journal of Industrial and Management Optimization*, 16 (2):857–885, 2020.

[28] P. Mitropoulos, I. Mitropoulos, I. Giannikos, and A. Sissouras. A biobjective model for the locational planning of hospitals and health centers. *Health Care Management Science*, 9(2): 171–179, 2006.

[29] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100, 1997.

[30] M. Ndiaye and H. Alfares. Modeling health care facility location for moving population groups. *Computers & Operations Research*, 35(7):2154–2161, 2008.

[31] I.H. Osman and S. Ahmadi. Guided construction search metaheuristics for the capacitated p-median problem with single source constraint. *Journal of the Operational Research Society*, 58(1):100–114, 2007.

[32] I.H. Osman and N. Christofides. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research*, 1(3):317–336, 1994.

[33] H. Qin, Y. Han, B. Zhang, L. Meng, Y. Liu, Q. Pan, and D. Gong. An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem. *Swarm and Evolutionary Computation*, 69:100992, 2022.

[34] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.

[35] S.S.R. Shariff, N.H. Moin, and M. Omar. Location allocation modeling for healthcare facility planning in Malaysia. *Computers & Industrial Engineering*, 62(4):1000–1010, 2012.

[36] H. K. Smith, P. R. Harper, C. N. Potts, and A. Thyle. Planning sustainable community health schemes in rural areas of developing countries. *European Journal of Operational Research*, 193 (3):768–777, 2009.

[37] S. Taymaz, C. Iyigun, Z. P. Bayindir, and N. P. Dellaert. A healthcare facility location problem for a multi-disease, multi-service environment under risk aversion. *Socio-Economic Planning Sciences*, 71:100755, 2020.

[38] W. Zhang, K. Cao, S. Liu, and B. Huang. A multi-objective optimization approach for healthcare facility location-allocation problems in highly developed cities such as Hong Kong. *Computers, Environment and Urban Systems*, 59:220–230, 2016.

[39] W. Zou, J. Zou, H. Sang, L. Meng, and Q. Pan. An effective population-based iterated greedy algorithm for solving the multi-AGV scheduling problem with unloading safety detection. *Information Sciences*, 657:119949, 2024.

Appendix

Definition of auxiliary working parameters

The auxiliary working parameter in the metaheuristic are the following:

d_{jl} Distance from site j to site l.
$\tilde{w}_{ki}$ Number of unallocated demand of institution k in demand point i; $k \in K$, $i \in N$.
λ_{ki} Distance from demand of institution k in origin i to the allocated site; $k \in K$, $i \in N$.
C_j Residual capacity in site j; $j \in M$.
CI_j Residual capacity for the demand of other institutions in site j; $j \in M$.
$\tilde{V}_j$ Maximum number of additional kernels that can be installed site j; $j \in M$.
O_k Maximum number of demand that can be assigned to other institutions; $k \in K$.
$\tilde{G}_k$ Number of unassigned kernels of institution k; $k \in K$.
$\tilde{P}_k$ Maximum number of CS that can be selected for institution k; $k \in K$.

The following working parameters must be calculated in the procedures of the metaheuristic. The distance for each demand point (k, i) to the allocated active site j is calculated as follows:

$$\lambda_{ki} = \begin{cases} d_{ij} & \text{if} \quad \tilde{x}_{ki} = j \\ D_0 & \text{otherwise} \end{cases} \tag{26}$$

The number of kernels to add for each $j \in CL$ in Pseudo-code 4 is determined as follows:

$$u_j^1(\Phi) = \begin{cases} \max\{H_j, 1\} & \text{for strategy } \Phi = \text{(i)} \\ \min\{\tilde{V}_j, \tilde{G}_{k(j)}\} & \text{for strategy } \Phi = \text{(ii)} \\ \max\{H_j, \min\{\lceil \tilde{V}_j/2 \rceil, \tilde{G}_{k(j)}\}\} & \text{for strategy } \Phi = \text{(iii)} \end{cases} \tag{27}$$

The number of kernels to add for each $j \in CL$ in Pseudo-code 6 are the following:

$$u_j^2(\Phi) = \begin{cases} 1 & \text{for strategy } \Phi = \text{(i)} \\ \min\{\tilde{V}_j, \tilde{G}_{k(j)}\} & \text{for strategy } \Phi = \text{(ii)} \\ \min\{\lceil \tilde{V}_j/2 \rceil, \tilde{G}_{k(j)}\} & \text{for strategy } \Phi = \text{(iii)} \end{cases} \tag{28}$$

The number of kernels to add in the active site j in pseudo-code 8 is calculated as follows:

$$u_j^3 = \max\{H_j, 1\} \tag{29}$$

The potential capacity for each $j \in CL$ according to $u_j^p(\Phi)$ is the following:

$$PC_j = KC \times u_j^p(\Phi) \qquad p = 1, 2, 3 \tag{30}$$

The benefit in the objective function of each element of the candidate list is calculated as follows:

$$\delta_j = \sum_{k \in K} \sum_{i \in N | PC_j \geqslant \tilde{w}_{ki}} \max\{\tilde{w}_{ki}(\lambda_{ki} - d_{ij}), 0\} \tag{31}$$

The benefit of allocating the demand (r, i) to the active site j is computed as follows:

$$\theta_{ki} = \max\{w_{ki}(\lambda_{ki} - d_{ij}), 0\} \tag{32}$$

Pseudo-code of local search strategies

The Pseudo-code of LS1 is the following:

Pseudocode 13 First-improvement local search for the interchange of sites

1: **procedure** LS1($\mathcal{S}$, R, *time_limit*, D_1)
2: $\mathcal{S}^* \leftarrow \mathcal{S}$;
3: $\mathcal{S}_0 \leftarrow \mathcal{S}$;
4: **while** (Improvement = true **and** time < *time_limit*) **do**
5: select solution $\mathcal{S}$;
6: $CL_1 \leftarrow \{j \in M_B | \tilde{y}_j = 1\}$;
7: **while** $(CL_1 \neq \emptyset)$ **do**
8: select an element $j_1 \in CL_1$;
9: $CL_2 \leftarrow \{j \in M_{B_{k(j_1)}} | \tilde{Y}_j = 0,\ \tilde{v}_{j_1} \geq H_j,\ \tilde{v}_{j_1} \leq V_j\}$
10: $CL_2 \leftarrow$ the R nearest elements to j_1 from CL_2.
11: **while** $(CL_2 \neq \emptyset)$ **do**
12: select an element $j_2 \in CL_2$;
13: update working parameters: $\tilde{V}_{j_1} \leftarrow \tilde{V}_{j_1} + \tilde{v}_{j_1}$, $\tilde{V}_{j_2} \leftarrow \tilde{V}_{j_2} - \tilde{v}_{j_1}$;
14: update solution: $\tilde{v}_{j_2} \leftarrow \tilde{v}_{j_1}$, $\tilde{v}_{j_1} \leftarrow 0$, $\tilde{y}_{j_1} \leftarrow 0$, $\tilde{y}_{j_2} \leftarrow 1$;
15: $CL_3 \leftarrow \{j_1, j_2\}$
16: $\tilde{x} \leftarrow$ DEMAND_DEALLOCATION($\mathcal{S}$, CL_3, D_1);
17: $\tilde{x} \leftarrow$ DEMAND_ALLOCATION($\mathcal{S}$);
18: **if** $(Z(\mathcal{S}) < Z(\mathcal{S}^*))$ **then**
19: $\mathcal{S}^* \leftarrow \mathcal{S}$;
20: $\mathcal{S}^0 \leftarrow \mathcal{S}$;
21: go to Step 5;
22: **else**
23: $\mathcal{S} \leftarrow \mathcal{S}_0$;
24: **end if**
25: $CL_2 \leftarrow CL_2 \backslash \{j_2\}$;
26: **end while**
27: $CL_1 \leftarrow CL_1 \backslash \{j_1\}$;
28: **if** $(CL_1 = \emptyset)$ **then**
29: Improvement = false;
30: **end if**
31: **end while**
32: **end while**
33: **return** $(\mathcal{S}^*)$
34: **end procedure**

The Pseudo-code of LS2 is the following:

Pseudocode 14 Local search for the interchange of new capacity

1: **procedure** LS2($\mathcal{S}$, R, $time\_limit$, D_1)
2: $\mathcal{S}^* \leftarrow \mathcal{S}$;
3: $\mathcal{S}_0 \leftarrow \mathcal{S}$;
4: **while** (Improvement = true **and** time $<$ $time\_limit$) **do**
5: select solution $\mathcal{S}$;
6: $CL_1 \leftarrow \{j \in M_A | \tilde{v}_j > 0\}$;
7: **while** ($CL_1 \neq \emptyset$) **do**
8: select an element $j_1 \in CL_1$;
9: $CL_2 \leftarrow \{j \in M_{A_{k(j_1)}} | \tilde{V}_j > 0\}$
10: $CL_2 \leftarrow$ the R nearest elements to j_1 from CL_2.
11: **while** ($CL2 \neq \emptyset$) **do**
12: select an element $j_2 \in CL_2$;
13: update working parameters:
let be $u \leftarrow \min\{\tilde{v}_{j_1}, \tilde{V}_{j_2}\}$, $\tilde{V}_{j_1} \leftarrow \tilde{V}_{j_1} + u$, $\tilde{V}_{j_2} \leftarrow \tilde{V}_{j_2} - u$;
14: update solution: $\tilde{v}_{j_2} \leftarrow \tilde{v}_{j_2} + u$, $\tilde{v}_{j_1} \leftarrow \tilde{v}_{j_1} - u$;
15: $CL_3 \leftarrow \{j_1, j_2\}$
16: $\tilde{x} \leftarrow$ DEMAND_DEALLOCATION($\mathcal{S}$, CL_3, D_1);
17: $\tilde{x} \leftarrow$ DEMAND_ALLOCATION($\mathcal{S}$);
18: **if** ($Z(\mathcal{S}) < Z(\mathcal{S}^*)$) **then**
19: $\mathcal{S}^* \leftarrow \mathcal{S}$;
20: $\mathcal{S}^0 \leftarrow \mathcal{S}$;
21: go to Step 5;
22: **else**
23: $\mathcal{S} \leftarrow \mathcal{S}_0$;
24: **end if**
25: $CL_2 \leftarrow CL_2 \backslash \{j_2\}$;
26: **end while**
27: $CL_1 \leftarrow CL_1 \backslash \{j_1\}$;
28: **if** ($CL_1 = \emptyset$) **then**
29: Improvement = false;
30: **end if**
31: **end while**
32: **end while**
33: **return** ($\mathcal{S}^*$)
34: **end procedure**