Algoritmos metaheurísticos para el problema de ubicación de instalaciones P-Mediana no capacitado

Santiago José Guzmán Villamarin
Faber Duban Garcés Gómez

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR: DRA. MARTHA ELIANA MENDOZA BECERRA
CO-DIRECTOR: DR. CARLOS ALBERTO COBOS

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE SISTEMAS
GRUPO DE I+D EN TECNOLOGÍAS DE LA INFORMACIÓN (GTI)
LÍNEA INVESTIGACIÓN: SISTEMAS INTELIGENTES
POPAYÁN
Noviembre 2021

Algoritmos metaheurísticos para el problema de ubicación de instalaciones P-Mediana no capacitado

Santiago José Guzmán Villamarin
Faber Duban Garcés Gómez

DIRECTOR: DRA. MARTHA ELIANA MENDOZA BECERRA
CO-DIRECTOR: DR. CARLOS ALBERTO COBOS

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE SISTEMAS
GRUPO DE I+D EN TECNOLOGÍAS DE LA INFORMACIÓN (GTI)
LÍNEA INVESTIGACIÓN: SISTEMAS INTELIGENTES
POPAYÁN
Noviembre de 2021

Agradecimientos

Agradecemos principalmente a DIOS por darnos la vida y esta maravillosa oportunidad, a nuestros padres por creer siempre en nosotros y en nuestras capacidades obtenidas en el transcurso de esta carrera, a nuestras familias por el apoyo incondicional para salir adelante y así poder terminar de manera exitosa tanto la carrera como este proyecto.

A los doctores Martha Mendoza y Carlos Cobos, que, gracias a su paciencia, compromiso, empeño y destreza nos lograron trasmitir su alto grado de conocimiento, el cual hizo posible la culminación de esta gran etapa, enriqueciendo nuestra experiencia como futuros ingenieros. Así mismo, a todos los profesores que hicieron parte de este proceso de formación, mil gracias por su excelente trabajo como educadores.

Gracias también a todos nuestros compañeros que estuvieron con nosotros a lo largo de esta maravillosa etapa, por su amistad, su apoyo y por los agradables y grandiosos momentos que logramos compartir.

CONTENIDO

Lista de tablas

INDICE DE FIGURAS

Glosario

ABC	Algoritmo de abejas artificiales (Artificial Bee Colony)
FLP	Problemas de ubicación de instalaciones (FLP, Facility location problem)
FmedianRPE	Función de error porcentual relativo
GTI	Grupo de investigación y desarrollo en tecnologías de la Información de la Universidad del Cauca.
HMCR	Tasa de consideración de la memoria (Harmony memory consideration rate)
HMS	Tamaño de la memoria de armonías (Harmony memory size)
HSOS	Hibridación de organismos simbióticos (HSOS, Hybrid symbiotic organism search)
LS	Búsqueda local (Local search)
NP-Completo	Problemas no polinomiales de complejidad alta o complejas
OIA	Operador de inicialización aleatorio
OIC	Operador de inicialización con conocimiento
OR	Investigación de operaciones (Operations research)
ORA	Operador de reparación aleatorio
ORC	Operador de reparación con conocimiento
ORLIB	Conjunto de instancias de prueba para diferentes problemas relacionados a la temática de investigación de operaciones
PAR	Probabilidad Ajuste armonía
SBHS	Búsqueda armónica binaria simplificada (SBHS, Simplified binary harmony search)
SD	Desviación estándar (Standard Desviation)
TS	Algoritmo Búsqueda Tabú (TS, Tabu Search)

Presentación

La importancia de ubicar instalaciones tiene un papel notable en la actualidad, ya que muchas empresas de diferentes sectores (Públicas y privadas) comprenden las ventajas de ubicar una instalación exitosamente, y además las desventajas que puede proporcionar al localizarlas en un espacio geográfico no adecuado, dependiendo del tipo de instalación que se requiera, por ejemplo, en Beijing (China) se consideró los contextos socio-demográficos y socioculturales de un área geográfica seleccionada, para determinar la posible ubicación de estaciones de carga de vehículos eléctricos. El enfoque principal del uso de la tecnología para ubicar las instalaciones es reducir costos, mejorar la eficiencia, y el servicio al cliente. Además, permite las reducciones drásticas de apertura, así como una mejora en los niveles del servicio ofrecido por la instalación.

En este trabajo se proponen dos algoritmos para la ubicación de instalaciones basados en la metaheurística de Búsqueda armónica binaria simplificada (SBHS) y una hibridación de organismos simbióticos (HSOS), que presentan buenos resultados en el problema de la mochila. En este documento se describe el proceso de implementación y adaptación de estos algoritmos al problema de ubicación de instalaciones, además, la base teórica que lo soporta.

La evaluación de los dos algoritmos adaptados se realizó por medio de la función de error relativo y la desviación estándar; sobre algunos conjuntos de datos presentados en OR-Library[1]. Los resultados de la evaluación se compararon con otros algoritmos del estado del arte, sin embargo, ningún algoritmo obtuvo los mejores resultados en todas las medidas.

Este documento está organizado de la siguiente forma:

El capítulo 1 describe el problema abordado en este proyecto, su justificación, la propuesta solución para el problema planteado, los objetivos desarrollados y los resultados obtenidos con el desarrollo de esta investigación.

El capítulo 2 detalla los conceptos teóricos del problema de ubicación de instalaciones no capacitadas y de los dos algoritmos base SBHS y HSOS. Así como revisión sistemática realizada, especificando la descripción de la cadena de búsqueda, los recursos literarios, el proceso de selección de estudios, y finalmente, los métodos metaheurísticos encontrados para el problema de ubicación de instalaciones no capacitadas.

El capítulo 3 presenta el proceso de adaptación de los algoritmos base, el cual se realizó en dos ciclos. El primer ciclo corresponde a la adaptación de los algoritmos base SBHS y HSOS, el ajuste y configuración de parámetros, además de los resultados parciales al adaptar estos algoritmos. Y para el segundo ciclo se describe el proceso de adición de búsqueda local sobre los algoritmos adaptados en el anterior ciclo, con el ajuste y configuración de parámetros para cada algoritmo.

El capítulo 4 describe el proceso de implementación de los algoritmos SBHS y HSOS al problema de FLP, en donde se describe la representación de la solución, la función objetivo

[1] Instancias tomadas en la siguiente URL: http://people.brunel.ac.uk/~mastjjb/jeb/info.html

utilizada, y los algoritmos adaptados al problema de FLP, por último, su implementación y resultados finales al modificar los parámetros.

El capítulo 5 describe los pasos correspondientes a la descripción de los datos usados de OR-LIB, las métricas usadas para la evaluación, el afinamiento de parámetros de los algoritmos adaptados, los resultados obtenidos por los dos algoritmos adaptados frente a otros métodos del estado del arte, finalmente, el resultado de la comparación de los dos algoritmos descritos en los ciclos del proceso de adaptación.

El capítulo 6 presenta las conclusiones del proceso de adaptación y adición de la búsqueda local en las implementaciones base de estos algoritmos al problema de FLP y los resultados de las evaluaciones realizadas a las adaptaciones. Además, se exponen las recomendaciones para trabajos de investigación similares y probables trabajos futuros que surgen del presente proyecto. Finalmente, se presentan las referencias bibliográficas que se tomaron como referente en el desarrollo del proyecto.

Capítulo 1

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Los problemas de ubicación de instalaciones (FLP, Facility Location Problem) tienen un alto grado de complejidad (NP-Completo[2]) para encontrar una solución que se considere óptima, la cual consiste en ubicar un número de instalaciones en un espacio propuesto, teniendo como objetivo minimizar las instalaciones a ubicar. Para algunos FLP se debe tener en cuenta la capacidad de demanda que puede atender cada instalación [1], clasificándolos en FLP capacitados y no capacitados. También, existen diferentes modelos de estos problemas teniendo en cuenta el objetivo que se quiere optimizar [2], como: cobertura, mediana, centro, jerárquico, entre otros.

El modelo de mediana consiste en ubicar una o más instalaciones para minimizar la distancia entre la demanda y su establecimiento más cercano. Una variación de este modelo es el de *P*-mediana, debido a que existe una restricción que requiere ubicar un número P definido de instalaciones. Este modelo de *P*-mediana no capacitado se ha aplicado para resolver diferentes problemas de la vida real. En Beijing (China) se consideró los contextos socio-demográficos y socioculturales de un área geográfica seleccionada, para determinar la posible ubicación de estaciones de carga de vehículos eléctricos [3]. Posteriormente en Turquía se desarrolló una metodología para la ubicación de posibles estaciones para el préstamo de bicicletas de uso compartido en lugares públicos, considerando los puntos de demanda en el campus de la Universidad de Gaziantep [4]. Y recientemente en Reino Unido se propuso el uso de este modelo para asignar la demanda a instalaciones médicas que prestan servicios prenatales en un área geográfica definida [1].

Debido a la complejidad de los problemas de *P*-mediana no capacitado, se han aplicado enfoques metaheurísticos que hacen uso de la aleatoriedad para encontrar mejores soluciones. El algoritmo de colonia de abejas artificial es una metaheurística que se aplica a este problema y ha sido probado con 40 instancias de un conjunto de datos presentados en la librería OR-Library [5], obteniendo los mejores resultados en la mayoría de las instancias, frente a otros algoritmos planteados en la literatura.

Otra metaheurística que está siendo frecuentemente utilizada en los últimos años para resolver diferentes problemas de optimización se conoce como el algoritmo de Búsqueda Armónica [6], la cual está inspirada en la realización de nuevas armonías por parte de los músicos de jazz. Esta técnica se ha usado en otros modelos diferentes a *P*-mediana, como en [7] que propone una variación al algoritmo original, teniendo en cuenta más de un objetivo para ubicar instalaciones que tienen una capacidad limitada y que a su vez compiten por la demanda del mercado con otras instalaciones ya establecidas, obteniendo buenos resultados. En [8] consideran las emisiones de gases de efecto invernadero (GEI) generadas por el constante tráfico desde los puntos de demanda hacia las instalaciones

[2] NP-Completo se refiere a todos los problemas no determinísticos de tiempo polinomial.

como una variable que afecta la solución del problema, este planteamiento es probado haciendo uso de diferentes algoritmos de búsqueda armónica híbridos propuestos.

Por otra parte, algoritmos híbridos de búsqueda armónica se ha implementado para problemas de *P*-mediana capacitado, tales como en [9] donde aplican un enfoque de agrupación de armonías por medio de una búsqueda local diseñada especialmente para resolver este tipo de problema, presentando buenos resultados frente a las propuestas de la literatura, especialmente donde el tamaño de la dimensión aumenta. También, en [10] propone el uso de una búsqueda amónica codiciosa, diversificando las notas en la memoria armónica permitiendo encontrar mejores soluciones a los algoritmos en comparación, especialmente cuando el número de medianas no es alto.

Existen varios trabajos que demuestran que el algoritmo de búsqueda armónica es efectivo para solucionar problemas NP-Completo binarios como el de la Mochila [11]. A partir de esta premisa, teniendo en cuenta que FLP también se representa de forma binaria a través de una conversión, y que las instancias de OR-Library para ubicar instalaciones *P*-mediana oscilan entre 100 a 7200 vértices; se seleccionaron dos variantes desarrolladas de este algoritmo que han sido publicadas en los últimos cinco años, y que han dado menor desviación estándar para solucionar el problema de la mochila en este mismo rango de vértices.

Por lo tanto, en este proyecto se propone utilizar los siguientes algoritmos para el FLP: [12] una versión simplificada del algoritmo de búsqueda armónica y una propuesta de organismos simbióticos hibridado con búsqueda armónica [13]. Lo cual conlleva a la siguiente pregunta de investigación. ¿Qué modificaciones se deben efectuar a los algoritmos versión simplificada del algoritmo de búsqueda armónica y una propuesta de organismos simbióticos hibridados con búsqueda armónica de manera que se adapten al planteamiento del modelo para el problema de *P*-mediana no capacitado?

1.2 APORTES

El desarrollo de este proyecto busca contribuir con el conocimiento de dos nuevos algoritmos que permiten ubicar instalaciones haciendo uso del modelo de *P*-mediana. Estos dos algoritmos (Búsqueda armónica simplificada SBHS y una hibridación de organismos simbióticos HSOS) fueron adaptados al problema de ubicación de instalaciones, adicionando conocimiento en los operadores de inicialización y reparación. En la literatura no se encontró un reporte de que estos algoritmos hayan abordado este problema.

Además, la línea de investigación de Gestión de la Información y Sistemas Inteligentes del Grupo de I+D en Tecnologías de la Información contarán con el código fuente de estos algoritmos para futuras investigaciones en el área de ubicación de instalaciones.

1.3 OBJETIVOS

1.3.1 Objetivo general

Proponer los algoritmos metaheurísticos de búsqueda armónica simplificada y una hibridación de organismos simbióticos adaptados al problema de ubicación de instalaciones *P*-mediana no capacitados.

1.3.2 Objetivos específicos

1. Adaptar los algoritmos de búsqueda armónica simplificada y una hibridación de organismos simbióticos para el problema de ubicación de instalaciones, incluyendo conocimiento en la reparación, ajuste, fases naturales de los organismos y el operador de síntesis.

2. Incluir una búsqueda local en los dos algoritmos adaptados buscando una mayor explotación en las zonas de búsqueda.

3. Evaluar la función de error relativo y la desviación estándar de los dos algoritmos propuestos con respecto a otros del estado del arte haciendo uso de instancias presentadas en OR-Library.

1.4 RESULTADOS OBTENIDOS

Los productos obtenidos del desarrollo de este proyecto de investigación son:

- **Monografía del trabajo de grado**, presenta los conceptos teóricos necesarios para el desarrollo del proyecto, el proceso de adaptación e implementación a FLP de los algoritmos base, la presentación de los algoritmos adaptados, los resultados obtenidos sobre el conjunto de datos usados, y su comparación con otros métodos del estado del arte. Finalmente, las conclusiones, recomendaciones y el trabajo futuro que el GTI espera desarrollar con base en esta investigación.

- **Código de los algoritmos adaptados**, con el diagrama de clase, los atributos y métodos utilizados para la realización de este proyecto, y el diagrama de componentes realizado.

- **Articulo** donde se describen los algoritmos SBHS y HSOS adaptados al problema de FLP, la representación de las soluciones, la función objetivo, y las adaptaciones realizadas a los algoritmos para el problema de FLP no capacitado. Además, se presentan los resultados obtenidos con los dos algoritmos adaptados y la comparación con el estado del arte.

Capítulo 2

2 CONTEXTO TEÓRICO Y ESTADO DEL ARTE

En esta sección se define el contexto teórico para el desarrollo de este proyecto, además un mapeo sistemático para identificar los métodos metaheurísticos más relevantes que se han desarrollado para el problema de ubicación de instalaciones *P*-mediana no capacitados.

2.1 CONTEXTO TEÓRICO

En esta sección se presentan los conceptos básicos requeridos para abordar la solución de un problema FLP y la descripción de los algoritmos base utilizada en este proyecto.

2.1.1 Problemas de ubicación de instalaciones

Los problemas de ubicación de instalaciones (FLP) se encargan de ubicar un conjunto de establecimientos a fin de minimizar el costo de instalación y por ende satisfacer la demanda (o en su defecto una gran parte de esta), con respecto a un conjunto de restricciones. Esto es debido a que la decisión que se tome para ubicar una instalación en un espacio geográfico, es crítica en el planeamiento estratégico de las empresas [14].

Existen cuatro componentes que describen los FLP [14]:

- La demanda, la cual se asume que está ubicada en los puntos geográficos donde una instalación puede ser ubicada o en las rutas que conectan estos puntos.
- Las instalaciones, que serán ubicadas en un espacio geográfico donde se encuentra la demanda a satisfacer.
- Un espacio de búsqueda en donde las instalaciones pueden ser establecidas y los clientes ya están ubicados.
- Una métrica o estándar que indica la distancia o el tiempo entre la demanda y la instalación.

2.1.2 *P*-mediana

Los modelos de mediana son considerados como el objetivo principal en los problemas de ubicación, ya que el mayor número de problemas son modelados a través de este. Entre las aplicaciones más importantes se encuentran la ubicación de escuelas, hospitales, estaciones de radio, entre otras [14].

Los modelos de *P*-mediana buscan los puntos medios entre los espacios geográficos candidatos, permitiendo maximizar la demanda que será atendida únicamente por P instalaciones, y minimizar la función objetivo dada por la sumatoria de los costos (distancia y/o el volumen) [14].

Estos modelos están basado en las siguientes suposiciones: Una relación lineal entre el costo y la distancia, donde no se tiene en cuenta un periodo de tiempo, la capacidad de las instalaciones para atender la demanda no está limitada, no se tienen en cuenta los costos de apertura, y que a su vez deben ser de un mismo tipo de instalación, por último se tiene en cuenta que esta demanda se mantiene constante y que no se podrá desplazar o modificar la ubicación de la instalación [14].

El objetivo de este modelo se muestra en la Ecuación (*1*)

$$\min \sum_{i=1} \sum_{j=1} h_i d_{ij} X_{ij} \qquad i,j=1,2,\dots,n \tag{1}$$

Sujeto a:

$$\sum_{j} X_{ij} = 1 \tag{2}$$

$$\sum_{j} Y_j = P \tag{3}$$

$$X_{ij} \leq Y_j \tag{4}$$

$$X_{ij}, Y_j \in \{0,1\} \tag{5}$$

Donde: h_i es la demanda, d_{ij} es la distancia entre el nodo de demanda i y el nodo donde se puede establecer una instalación j (d_{ij} es cero si $i = j$), X_{ij} es la asignación de una demanda i a una instalación j, Y_j es la asignación de una instalación en un punto j, siendo i, j enteros de 1 hasta n y n el número total de puntos de demanda.

Esto está sujeto a las siguientes restricciones: la Ecuación (2) asegura que cada punto de demanda sea atendido solo por una instalación y se establece así, X_{ij} es igual a 1 si la demanda del nodo i esta asignada a una instalación en j, la Ecuación (3) restringe a P el número de instalaciones máximo a ubicar, en la Ecuación (4) se expresa la condición de no asignar un punto de demanda a una instalación que no esté abierta $Y_j = 0$ y por último en la Ecuación (5) se determina el dominio de las variables.

2.1.3 Algoritmos base

En esta sección se detallan los algoritmos base que fueron desarrollados en el problema de la mochila (QKP), así como los pasos más relevantes para el desarrollo de estos.

2.1.3.1 Búsqueda armónica simplificada

Es una versión simplificada del algoritmo de búsqueda armónica, donde se tiene en cuenta únicamente dos parámetros conocidos como el tamaño de la memoria y la tasa de consideración de la memoria de armonías [12]. El pseudocódigo de este algoritmo se presenta en la Figura 1 y se basa en los siguientes pasos:

- **Paso1 – Inicialización:** se basa en una probabilidad equitativa del 50% para la construcción de la memoria armónica (Línea 8), permitiendo tener en cuenta cada uno de los elementos dados para la solución, donde se ordena la memoria de forma ascendente de acuerdo con la función objetivo.

- **Paso2 - Reparación:** este procedimiento (Línea 9) hace uso de dos etapas codiciosas para reparar soluciones no viables expuestas en la memoria armónica inicial, donde cada elemento se ordenará teniendo en cuenta la densidad de este.

- **Paso3 - Improvisación:** Una nueva armonía se inicializa en 0 para cada una de sus dimensiones, la improvisación hace uso de dos reglas para crear una nueva armonía, las cuales son, la tasa de consideración (Línea 12) la cual es la probabilidad de generar un cambio en cada dimensión y un ajuste a esa solución (Línea 13), para lo cual se escogen dos individuos diferentes de la población de forma aleatoria ($r1$ y $r2$), además del mejor individuo con la finalidad de simplificar el proceso de escoger un elemento de la memoria así como un ajuste de este en una única fórmula. Por último, si la nueva armonía es mejor que la peor de la memoria, esta reemplaza la última de la memoria (Línea 20).

  Luego se repiten los pasos 2 y 3, hasta cumplir el criterio de parada (Línea 10), retornando la primera armonía de la memoria.

```
1  INICIO SBHS()
3  //Inicializacion de parámetros
4  HMCR = 1 – (1/numeroItems) //Tasa de consideración de la memoria en función del
5                          //Número de dimensiones del problema
6  Efos = 0     //Número de evaluaciones de la función objetivo.
7  MaxEfos     //Máximo número de evaluaciones de la función objetivo.
8  poplacion = InicializarPoblacion()
9  Reparar(poblacion)
10 MIENTRAS Efos < MaxEfos HACER
11   DESDE i = 0 hasta numeroItems HACER
12     SI aleatorio(0,1) < HMCR ENTONCES
13       estado = poblacion[r1].Items[i] +((-1)^(poblacion[r1].Items[i])) *
14 |poblacion[Best].Items[i]- poblacion[r2].Items[i]|
15     FIN SI
16     SI estado == 1 ENTONCES
17        nuevaArmonia[i] = estado
18     FIN SI
19   FIN DESDE
20   SI (Evaluar(nuevaArmonia, peor(poblacion))) ENTONCES
21      peor = nuevaArmonia
22   FIN SI
23 FIN MIENTRAS
24 FIN
```

Figura 1. Pseudocódigo base SBHS

2.1.3.2 Algoritmo hibrido de organismos simbióticos

Es un algoritmo de hibridación codiciosa basado en la propuesta de Cheng y Prayogo [13], que simula la interacción simbiótica que los organismos usan para sobrevivir al ecosistema, la cual consta de tres fases: mutualismo, comensalismo y parasitismo. A esta propuesta se le adiciona una fase de búsqueda de armonía para ayudar a encontrar mejores soluciones [13], [12]. El pseudocódigo de este algoritmo se presenta en la Figura 2 y se basa en los siguientes pasos:

Paso 1: El algoritmo inicializa aleatoriamente la población de organismos (Líneas 11 a 20) a partir de una probabilidad equitativa del 50% en el ecosistema a partir de unos parámetros necesarios para el funcionamiento del algoritmo (Líneas 2 a 8).

Paso 2: Posteriormente se identifica al mejor individuo de la población (Línea 24), de acuerdo con el objetivo del problema. Después, por cada individuo de la población X_i (Línea 25) se escoge un individuo de forma aleatoria X_j (Línea 27) que debe ser diferente al organismo X_j en cada una de las fases mencionadas en los siguientes pasos (Línea 30 y 33).

Paso 3 - Mutualismo: En esta fase los dos individuos X_i y X_j de la población entablan una relación de mutuo beneficio con el objetivo de aumentar las posibilidades de supervivencia (Línea 28 y 29), esta relación se realiza mediante la selección de que elemento puede ofrecer un mejor valor en la función objetivo haciendo uso de un operador de síntesis.

Paso 4 - Comensalismo: En esta fase, un solo organismo obtiene beneficio de otro, pero no genera ninguna modificación en este (Línea 32). Un organismo X_i mejora los elementos que contiene, a partir de las dimensiones que ofrecen un mejor valor en la función objetivo presentes en el organismo X_j.

Paso 5 - Parasitismo: En esta fase, un nuevo organismo al cual se llamará parasito se genera tomando elementos del organismo X_i (Línea 35) y si el parasito obtiene un mejor valor en la función objetivo al presentado por X_j, este lo reemplazara.

Paso 6 - Armonía: En la fase de armonía se realiza un proceso de transformación, en donde de toma los elementos a partir de la memoria de armonías (Línea 36) y de una probabilidad aleatoria equitativa, permitiendo que esta transformación del organismo tome valores aleatorios.

Paso 7: Por último, el algoritmo regresa al paso dos (2) realizando varias iteraciones sobre estas fases, hasta que se cumpla el criterio de finalización como el número máximo de iteraciones $MaxIter$ (Línea 22).

Paso 8: Al finalizar el número máximo de iteraciones (Línea 40) el algoritmo retorna a la mejor solución (organismo) y su evaluación de acuerdo con el problema.

Es importante resaltar que en los pasos del 3 al 6, en cada una de las fases se realiza un proceso de reparación adaptado al problema, este método busca que elemento aporta un mayor valor al ser insertado en la mochila, pero si este supera la capacidad que puede tener la solución, es eliminado.

```
INICIO
    // Inicializar los parámetros
        HMCR=0.5      //Tasa de consideración de la memoria
        PAR=0.365     //Probabilidad de ajuste
        Efos=0        //Número de iteración a ejecutarse
        MaxEfos=300   //Número máximo de iteraciones
        n=10          //Número de dimensiones para el problema
        M=5           //Tamaño de la población

     //inicializar el ecosistema de organismos
    DESDE i = 1 HASTA M HACER
        DESDE j = 1 HASTA n HACER
              xij = aleatorio(0, 1);
              SI xij < 0.5
                 xij = 0;
              SI NO
                 xij = 1;
              FIN SI
        FIN DESDE
    FIN DESDE
    //iteración en busca de mejores soluciones
    MIENTRAS Efos < MaxEfos
        DESDE i=1 HASTA M HACER
              XMejor = mejorPoblacion()
              Xi = M[i]
              j= aleatorio(i)
              Xj =M[j]
              mutualismo(Xi, Xj, XMejor)
              mutualismo (Xj, Xi, XMejor)
              j= aleatorio(i)
              Xj =M[j]
              comensalismo (Xi, Xj, XMejor)
              j= aleatorio(i)
              Xj =M[j]
              parasitismo (Xj, Xi)
              Armonia(Xi)
       FIN DESDE
       Efos=Efos+1
    FIN MINETRAS
  Xmejor = mejorPoblacion()
FIN
```

Figura 2. Pseudocódigo base HSOS

2.2 ESTADO DEL ARTE

Para el desarrollo del estado del arte se realizó una revisión sistemática de los métodos de metaheurística para los problemas de ubicación de instalaciones no capacitadas, siguiendo algunos lineamientos bases planteados en [15], que ayudan a clasificar diferentes artículos de acuerdo con las preguntas y a la motivación para el estado del arte, estas preguntas se detallan en la Tabla 1.

Tabla 1 Preguntas y motivación para el estado del arte

ID	Pregunta	Motivación
P1	¿Qué trabajos de investigación enfocados al problema de ubicación de instalaciones *P*-mediana se han desarrollado?	Revisar el estado del arte actual sobre este problema
P2	¿Qué algoritmos metaheurísticos han obtenido buenos resultados para solucionar el problema de ubicación de instalaciones *P*-mediana?	Reconocer que algoritmos metaheurísticos se han usado para resolver este problema.
P3	¿Qué algoritmos de búsqueda local han obtenido buenos resultados para solucionar el problema de ubicación de instalaciones *P*-mediana?	Reconocer que algoritmos de búsqueda local se han usado para resolver este problema.

2.2.1 Cadenas de búsqueda

Con el fin de conocer los avances en investigación en el área de ubicación de instalaciones no capacitadas, y tener un punto de partida para este trabajo, se generaron las cadenas de búsqueda que permitieron encontrar todos los documentos relacionados en un rango de tiempo específico. Estas cadenas de búsqueda están conformadas por los términos más relevantes para el tema abordado, como se muestra en la Tabla 2

Tabla 2 Cadenas de búsqueda

Pregunta	Cadena de búsqueda
P1	(“State of art” OR “Survey” OR “Review”) **AND** (“Facility Location Problems” OR “Facility Location Models”)
P2	(“Facility Location Problems” OR “Facility Location Models”) **AND** (“P-Median”) **AND** (“Algorithms” OR “Methods” OR “Techniques”)
P3	(“Binary Harmony Search”) **AND** (“Local Search Algorithms”)

2.2.2 Recursos literarios

Las cadenas de búsquedas presentadas en la sección anterior fueron ejecutadas en las siguientes bases de datos o metabuscadores:

- IEEE Digital library.
- Scopus.
- Elsevier.
- ScienceDirect.
- Springer.
- ACM.

2.2.3 Proceso de selección de estudios

Este proceso se llevó a cabo basándose en el título, palabras clave y resumen (abstract). Para la selección de estudios más relevantes se utilizaron los criterios de Inclusión (CI) y Exclusión (CE), definidos a continuación:

CI1: El trabajo debe haber sido publicado desde el 2015 hasta la fecha.

CI2: Los algoritmos de búsqueda local han sido implementados en algoritmos de búsqueda armónica binaria.

CE1: El trabajo no propone un método para la ubicación de instalaciones *P*-mediana no capacitado.

2.2.4 Métodos metaheurísticos para la ubicación de instalaciones no capacitada

En la revisión de literatura se encontraron los siguientes artículos que usan métodos metaheurísticos para abordar el problema de ubicación de instalaciones no capacitadas. Estos por lo general hacen uso de una representación de la solución binaria donde 1 indica que esa ubicación (dentro de los posibles puntos de ubicación) se localiza una instalación y 0 en el caso contrario, donde no se ubica ninguna instalación en este punto. En algunos artículos hacen uso de la representación continua para la evolución del algoritmo, siendo necesario una codificación al inicio del algoritmo y una decodificación al final para evaluar la función objetivo.

En 2015 se propone un algoritmo genético modificado para el problema de ubicación de instalaciones [16] de la siguiente forma: la asignación de las posibles ubicaciones como un gen, inicializando la población con soluciones factibles a partir de un sistema no aleatorio, luego se aplica un cruce a la mitad de la población, por último después de diez iteraciones de ejecución de este algoritmo se adiciona una nueva etapa que incluye dos “inmigrantes” en la población. En [5] utilizan las instancias de OR_LIBRARY con el algoritmo de colonia de abejas artificial continuo, en el cual ordenan de menor a mayor las distancias euclidianas de la instalación a su demanda respectiva. Además, realizan una conversión a la solución discreta que entrega el algoritmo a una binaria, al mismo tiempo que realiza una reparación garantizando que la solución sea factible, para finalmente realizar un proceso de explotación con el propósito de mejorar la calidad de esta.

En 2016 [17] propone el uso de una función Lagrangiana y semi-Lagrangiana optimizadas con un método de ascenso dual, para ser aplicadas en problemas de ubicación de instalaciones de *P*-mediana, permitiendo reducir el tiempo de ejecución frente a otros algoritmos presentados y por ende la obtención de una solución para encontrar multiplicadores óptimos, y en [18] hacen uso de una mejora de los algoritmos DROP y ADD, usando la teoría de grafos para optimizar los servicios postales en Italia, en el cual modifican los límites superiores e inferiores de la función objetivo para ir acotando el espacio de búsqueda hasta encontrar la solución esperada.

En 2018 Karatas [19] presenta una metodología para resolver problemas de ubicación de instalaciones multiobjetivo centradas en estaciones de servicios de emergencia haciendo uso del algoritmo interfloc, adicionando nuevas restricciones al modelo que buscan maximizar la demanda que ya está cubierta, restringiendo el número P de instalaciones que han sido ubicadas anteriormente. Además [20] propone el uso del algoritmo de temple simulado como un algoritmo de búsqueda local, que permite encontrar soluciones cercanas al límite superior dentro del espacio de búsqueda en el modelo de *P*-mediana para problemas de ubicación.

Posteriormente, en 2019 [1] hacen uso de un algoritmo llamado búsqueda tabú el cual integra estructuras de memoria que penalizan a la solución cuando la distancia es mayor a la esperada y así evitar los ciclos en el proceso de exploración y evalúan con diferentes instancias de la base de datos OR-Library. Además, en [21] modifican el algoritmo de

hormigas artificiales realizando un ajuste en los parámetros para convertirlo en una colonia de hormigas artificiales, que consta de dos etapas para ubicar las instalaciones, en la primera etapa maximizan su capacidad de demanda, y en la segunda mejoran la solución obtenida teniendo en cuenta el presupuesto restante para abrir una instalación, permitiendo aplicarlo a una formulación de una variación del modelo con demanda elástica, donde la demanda se toma como una variable que está cambiando en el tiempo, y por último [22] presenta un algoritmo de aprendizaje evolutivo para diferentes modelos de ubicación de instalaciones, para el cual, en la fase de selección elitista, se escogen la mejor y peor solución de la población permitiendo que el algoritmo converja a una solución cercana del óptimo.

Recientemente, en 2020 [23] crean un método a partir de árboles binarios de búsqueda para identificar la ubicación de las instalaciones donde el primer nodo representa la solución óptima conocida y los posteriores a variaciones de la solución al hacer un movimiento, y en [24] aplican el algoritmo de búsqueda tabú, en donde agregan dos listas para identificar que instalaciones se han añadido o eliminado de una solución para no volverlas a usar, estas listas castigan las soluciones creadas, a partir de movimientos de eliminación o adición para posteriormente ser evaluadas, si una solución de alguna de las listas es mejor que la actual, esta es reemplazada.

Por último, en 2021 [25] hace una investigación sobre los algoritmos más usados para resolver el problema de *P*_mediana no capacitado, con los parámetros necesarios para la ejecución de estos, y los métodos de búsqueda local que más se adaptan al problema. Estos algoritmos son métodos de cruce son: (1) eliminación codiciosa [26], el cual genera una nueva solución realizando un cruce entre las dimensiones de 2 individuos de la población de forma aleatoria, este procedimiento se realiza de la siguiente forma: si el valor obtenido en las dimensiones de los 2 individuos es igual, este valor es igual para la nueva solución, si el valor obtenido en las dimensiones es diferente, toma el valor de la solución en donde se ha ubicado una instalación, y repara esto obteniendo la mínima perdida en la evaluación de la función objetivo; (2) adición y eliminación codiciosa [27] que agrega un nuevo método de búsqueda que tiene como objetivo obtener un mayor valor al adicionar una instalación sobre la solución; (3) intercambio [28] que busca cambiar los valores en las dimensiones que tienen 2 individuos de la población, realizando sobre estos un número determinado de reemplazos de acuerdo con un parámetro establecido; este intercambio se realiza cuando el valor de la dimensión en una solución sea diferente a la otra, y posteriormente esta solución es reparada obteniendo la mínima perdida en la evaluación de la función objetivo.

De igual forma, el uso de algoritmos de búsqueda local han sido implementados sobre esta problemática para dar una mayor explotación al hacer uso de las metaheurísticas, en [5] desarrollan este tipo de algoritmo para modificar de forma secuencial las dimensiones de la solución dada por la metaheurística, aplicando una forma exhaustiva para la cual cada elemento representado en la solución que es una instalación es modificada, evaluando los que no representan una instalación, comparando esta con la solución actual hasta encontrar un mejor resultado.

En [20] aplican el algoritmo de temple simulado, el cual parte de una solución inicial con un parámetro de temperatura específica (L), para cada valor de temperatura se ejecuta un número de iteraciones determinado y en cada uno de ellas se instancian varios vecinos. Cada uno de estos vecinos se generan a partir de una variable de proyección, la cual indica que instalación se va a cerrar y cuál de estas se va a abrir sobre la solución actual. Esta solución es aceptada como un nuevo individuo en la población dependiendo de la

probabilidad $p = \exp\{-\frac{\alpha}{t}\}$. Este proceso continúa mientras que la temperatura está bajando, y finaliza cuando entra a un estado de congelación $L = 0$.

En [29] aplican un método exhaustivo donde se toma la lista de posibles puntos para ubicar una instalación, se compara y se selecciona el espacio geográfico que da un mejor valor en la evaluación de la función objetivo para ubicar la instalación lo más lejano posible. En caso contrario, se busca la mayor distancia que se puede obtener al ubicar esta instalación teniendo en cuenta ciertos criterios como la distancia en la función objetivo.

En [30] adaptan el algoritmo de la araña social al problema de *P*_mediana no capacitado, para lo cual crean espacios binarios de forma aleatoria, y hacen que una araña genere nuevas soluciones a partir de la dirección de movimiento en una anterior iteración, si encuentra un mejor resultado esta cambia a partir de su estado.

Capítulo 3

3 PROCESO ADAPTACION: ALGORITMOS SBHS y HSOS

Para el desarrollo de este proyecto se tomó como guía el Patrón de Investigación Iterativa (PII) propuesto por Pratt [31], el cual ha sido creado para proyectos de investigación que involucran una solución computacional. Este propone ciclos compuestos de cuatro etapas principales: observación, identificación del problema, desarrollo de la solución y prueba de la solución; y al final del ciclo se entregarán unos productos específicos.

Para realizar la adaptación y adición de los algoritmos SBHS y HSOS al problema de ubicación de instalaciones, se realizaron dos ciclos iterativos. Además, una etapa de documentación y divulgación realizada de forma paralela al proyecto.

3.1 CICLO I: ADAPTACION DE ALGORITMOS SBHS Y HSOS

En esta sección se presenta la función objetivo del problema de ubicación de instalaciones *P*_mediana no capacitada, la definición de los parámetros iniciales de los algoritmos SBHS y HSOS, el algoritmo de Floyd adaptado al problema, diferentes métodos de inicialización y reparación de las soluciones. Por último, se realizó el afinamiento de parámetros de los algoritmos adaptados.

3.1.1 Función objetivo ubicación de instalaciones *P*_mediana no capacitado

La función objetivo del problema de ubicación de instalaciones (FLP) no capacitado, tiene en cuenta características que ayuda a identificar las posibles ubicaciones en el espacio geográfico, asignándole la mínima distancia. El conjunto de características usado para la configuración base de la función objetivo ecuación (6), es el reportado en [5], considerando los buenos resultados obtenidos.

$$f_i^k = f\left(S_i^k\right) = \sum_{i=1}^{NP} \left(\min\left(\text{dist[i]}\left[m_j\right]\right)\right) \tag{6}$$

$$m_j \in \{m_1, m_2, m_3, \dots, m_p$$

Dónde: f_i^kes el valor de la función objetivo de una solución i perteneciente a una población de tamaño k. Igualada a la sumatoria de la mínima distancia entre un punto de demanda i hacia las instalaciones m_j. Teniendo en cuenta que la función tiene una restricción sobre el número de instalaciones P que serán ubicadas en el espacio geográfico.

3.1.1.1 Distancia

Se define como distancia al recorrido de viajar entre los puntos de demanda hacia las instalaciones, considerándose desde una perspectiva línea a la longitud del segmento de la recta que los une.

3.1.1.2 Asignación

Se conoce como asignación a los puntos de demanda i que se asignan a una instalación m_j, para lo cual j varía entre 1 al número de instalaciones p a ubicar en el problema.

3.1.2 Parámetros iniciales

Para definir de manera preliminar los parámetros de los algoritmos SBHS y HSOS adaptados a FLP no capacitado, se tomó como referencia la evaluación y análisis de los parámetros como un problema discreto [12],[13], adaptándolos al problema de FLP.

3.1.2.1 HSOS

El algoritmo HSOS utiliza el siguiente parámetro:

- **Tasa de ajuste de tono** (PAR): Define el ajuste o modificación que se realizará a cada dimensión de la armonía, el cual es 0.365.

3.1.2.2 Parámetros que usan ambos algoritmos

Los algoritmos SBHS y HSOS utilizan los siguientes parámetros:

- **Tasa de consideración de la memoria armónica** (HMCR): Define la tasa de consideración de elementos de la memoria para la nueva armonía, se usa de manera preliminar la ecuación (7).

$$HMCR = (1 - (10/\mathrm{NP})) \tag{7}$$

  Donde NP es un parámetro asociado al problema.

- **Tamaño de la memoria armónica** (HMS): Es el tamaño que se define para la población, es decir, la memoria armónica, el cual es de 30.

- **Número máximo de evaluaciones de la función objetivo** (maxEFOS): Es el número máximo de evaluaciones que se ejecutarán, en caso de que los algoritmos adaptados no encuentren la solución óptima. Este valor es de 1000 evaluaciones

3.1.2.3 Asociados al problema

En cuanto a los parámetros asociados al problema, se toma en cuenta maxEFOS, esto es debido a que los algoritmos pueden converger en una solución u óptimo local. Además, los siguientes parámetros que se definen en cada conjunto de datos de OR-LIB:

- **Número de posibles ubicaciones (NP):** Número de posibles puntos de localización donde una instalación puede ser ubicada.

- **Número de instalaciones a ubicar (*P*):** Número de instalaciones que serán ubicadas en el espacio geográfico.

- **Óptimo conocido (OptKnow):** Valor óptimo al ubicar P instalaciones en el espacio geográfico.

Obteniendo como resultado, los parámetros iniciales que se observan en la Tabla 3.

Tabla 3 Parámetros Iniciales

Parámetro		Valor
SBHS	Tamaño de la memoria Tasa de consideración	30 (1-(1/NumVertices))
HSOS	Tamaño de la memoria Probabilidad ajuste memoria Tasa de consideracíon	30 0.365 (1-(1/NumVertices))

3.1.3 Algoritmo de Floyd

Debido a la estructura de los conjuntos de datos de la librería OR-LIB usados en este proyecto, para el problema de *P*-mediana no capacitado, Beasley [5] sugiere el uso del algoritmo de Floyd sobre los datos para hacer un grafo no dirigido sobre ellos, permitiendo determinar las distancias mínimas entre un punto de demanda hacia una posible instalación.

Este algoritmo para encontrar el camino mínimo, busca el camino entre todos los pares de vértices en una única ejecución, comparando todas las rutas posibles a través de la matriz de distancias como la presentada en la Figura 3. La matriz de Floyd parte del hecho que hay diferentes puntos conectados entre sí, si el punto no encuentra una conexión con otro, lo indica en la matriz como un valor muy elevado o infinito.

$$M = \begin{bmatrix} 0 & 8 & 5 & \infty & \infty \\ \infty & 0 & 12 & \infty & \infty \\ 5 & 12 & 0 & \infty & 15 \\ 10 & \infty & 2 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

Figura 3. Representación inicial matriz distancias

Posteriormente el algoritmo mejora paulatinamente cada punto que esta interconectado en el grafo por medio de estimaciones para hallar el camino más corto, hasta conocer que la estimación es óptima. Como se observa en la Figura 3, el primer punto de demanda (fila y columna 1), no está conectado directamente al último punto (fila 1, columna 5), por lo cual aparece en la matriz como infinito. Posteriormente se estima como llegar a este punto haciendo la suma de distancias desde el primer punto de demanda al tercero (con valor de distancia de 5), y luego del tercero al quinto (con un valor de 15), logrando estimar que el primer punto está conectado al último con una distancia de 20, como se observa en la Figura 4.

$$M = \begin{bmatrix} 0 & 8 & 5 & \infty & 20 \\ 17 & 0 & 12 & \infty & 27 \\ 5 & 12 & 0 & \infty & 15 \\ 7 & 14 & 2 & 0 & 17 \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

Figura 4. Matriz Adyacencia aplicando algoritmo Floyd

La implementación de este algoritmo se realizó teniendo en cuenta las sugerencias propuestas por Beaskley [5], para tener una matriz completa de distancias mínimas de asignación. Para esta matriz, en la ecuación (8), se establece que cuando el punto i no tiene conexión con el punto j, el valor de la distancia es igual a $infinito$.

$$c(i,j) = \text{infinito} \;\; para \;\; i = 1, \ldots . n \; y \;\; j = 1, \ldots \ldots, n, i \neq j. \tag{8}$$

Donde n es el número de posibles ubicaciones en el espacio geográfico, i la asignación de un punto en la fila de la matriz, y j la asignación de un punto en la columna de la matriz.

Cuando la distancia de recorrer hacia el mismo punto (donde i sea igual a j), el valor de la distancia será igual a 0, tal como se muestra en la ecuación (9).

$$c(i,j) = 0 \; para \; i = 1, \ldots, n, i = j \tag{9}$$

Posteriormente se asigna un valor en la matriz de distancias igual a k, cuando se desea recorrer desde el punto i hacia una instalación j (y que i sea diferente a j), de este mismo modo, se asigna el mismo valor k cuando se requiere recorrer desde el mismo punto j hacia i, somo se muestra en la ecuación (10).

$$c(i,j) = k \; y \; c(j,i) = k \tag{10}$$

3.1.4 Inicialización

Para la inicialización de la población de los algoritmos HSOS y SBHS para el problema de la mochila, se usa la ecuación (11), que determina el número de elementos que serán insertados en la mochila (colocando el estado en 1) cuando un aleatorio es mayor o igual a 0.5, y cero si no se cumple la condición. Permitiendo obtener una probabilidad de 50% en la inserción de cada elemento en la mochila.

$$Random = \begin{cases} 1, si \; myRandom \geq 0.5 \\ 0, si \; myRandom < 0.5 \end{cases} \tag{11}$$

En el problema de la mochila no se realiza un proceso de reparación en las soluciones de la población inicial, debido a que busca maximizar el valor de elementos insertados en la mochila para que cada solución contenga una gran cantidad de elementos, y posteriormente en la evolución del algoritmo ser reparada.

Sin embargo, en el problema de *P*-mediana se busca ubicar solo P instalaciones y si no se hace reparación en la población inicial se generan para los dos algoritmos soluciones inviables como puntos de partida. Por esta razón, se buscó aplicar dos alternativas de inicialización en donde la reparación de cada solución haga la solución viable con respecto al problema.

3.1.4.1 Inicialización aleatoria controlada

Es un proceso en el cual se selecciona de forma aleatoria dentro del vector X_i un número de instalaciones igual a P. Para cada X_i se inicializa teniendo en cuenta el número P de instalaciones a ubicar. En la Tabla 4 se presentan dos ejemplos, donde n representa el número de vértices (puntos de demanda) donde puede ser ubicada una instalación, y P el número de instalaciones a ubicar.

Tabla 4. Inicialización aleatoria controlada ($n = 5$, $P = 2$)

X_i	1	2	3	4	5
X_1	0	1	1	0	0
X_2	1	1	0	0	0

La inicialización implementada (ver Figura 5) genera una lista de soluciones, la cual se llamará población (línea 2), para cada uno de los individuos se genera una lista de vértices de tamaño P, seleccionados de forma aleatoria (línea 5), luego se activa cada una de estos colocando un 1 en la posición donde se ubicará una instalación (línea 7), posteriormente esta solución es evaluada (Línea 8) y adicionada a la población (Línea 9).

```
1   INICIO
2         población = lista<solución>
3         PARA i = 0 HASTA tamañoPoblacion
4             solucion = nueva_solucion<solucion>
5             vertices = seleccionarPVerticesAleatoriamente(P)
6             PARA n = 1 HASTA vertices
7                 solución.activar(posiciones[n])
8             solución.evaluar()
9             poblacion.adicionar(solucion)
10        FIN PARA
11  FIN
```

Figura 5. Inicialización aleatoria controlada

3.1.4.2 Inicialización aleatoria con conocimiento

Para esta inicialización de la solución se tomó como base el planteamiento de los algoritmos base, realizando un cambio en el porcentaje para 1 y 0, (ver ecuación (12)), donde la probabilidad de ubicar una instalación es del 30%. El aleatorio ($myRandom$) es un número entre 0 y 1.0.

$$Random = \begin{cases} 0, si\ myRandom < 0.7 \\ 1\ si\ myRandom \geq 0.7 \end{cases} \quad (12)$$

La implementación de esta inicialización (ver Figura 6), al igual que la anterior crea una lista de soluciones llamada población, esta lista tiene un tamaño definido a partir del problema. Luego se itera sobre la lista de soluciones (línea 3), y a cada uno de sus vértices (línea 5) se les realiza una activación aleatoria (línea 7) de acuerdo con una probabilidad del 30%. Posteriormente, para que esta solución sea factible (número de instalaciones igual a P), se hace uso de un operador de reparación (línea 8), nombrado como reparador con conocimiento, el cual es detallado en la sección 3.1.5.1. Por último, se evalúa la solución (línea 9) y se adiciona la lista de la población (línea 10).

```
1   INICIO
2         población = lista<solución>
3         PARA i = 0 HASTA tamañoPoblacion
4             solucion = nueva_solucion<solucion>
5             PARA v = 0 HASTA solución.NumeroVertices
6                SI aleatorio() > 0.7
7                   solución.activar(v)
8             solución.reparar
9             solución.evaluar()
10            poblacion.adicionar(solucion)
11        FIN PARA
12  FIN
```

Figura 6 Inicialización aleatoria con conocimiento

3.1.5 Operador de Reparacion

Los algoritmos HSOS y SBHS para el problema de la mochila, cuentan con un operador de reparación. Debido al cambio en la función objetivo, se observa que en el problema de FLP *P*_mediana no capacitado no se puede hacer uso de estos operadores, esto es debido a que la reparación mencionada tiene en cuenta el valor de los objetos que serán almacenados en la mochila, sin importar el número de elementos que se encuentran en la solución, mientras que el problema FLP *P*_mediana no capacitado restringe el número de elementos a ubicar en una solución.

Por ende, se describen dos nuevos métodos para identificar si un proceso aleatorio o con conocimiento se adapta mejor a este tipo de problema, de los cuales uno está definido en la literatura para un problema similar al de ubicación de instalaciones *P*_mediana no capacitada.

3.1.5.1 Reparador con conocimiento

Este reparador se basa en el planteamiento del operador de reparación en [29], para soluciones del problema *P*_mediana desagradable, cuyo objetivo es ubicar instalaciones lo más lejos de los clientes, maximizando el valor de la función objetivo. Este operador parte del hecho de que una solución no cumple con la regla de ubicar un número P de instalaciones definidas, es decir, una solución puede presentar un número mayor o menor de estas al requerido en el espacio geográfico. También parte de que al agregar instalaciones el valor de la función objetivo disminuye y al eliminarlas el valor de la función objetivo aumenta, teniendo en cuenta lo anterior el reparador puede tomar dos casos:

- Primer caso. Cuando el número de instalaciones ubicadas es menor a P, para lo cual la ecuación (13) evalúa los puntos donde no se ha ubicado una instalación y mediante el uso de un vector de menores distancias, calculado a partir de la ecuación (14), determina para cada punto cuanto disminuye el valor la función al agregarse como instalación. La ecuación (15) selecciona el punto que genere una menor disminución dado que como se mencionó antes se está maximizando, posteriormente será agregado a la solución como una instalación.

$$PerdidaPorAdicion\ (j,x) = \sum_{i=0}^{i\in I} \min\{0, (d_{ij} - \delta_i)\}\ j\ \ni\ s_x \quad (13)$$

$$s_i = \min\{d_{ij}, \mathrm{j} \in s_x\} \quad (14)$$

$$k = \ Argmin\ \big(PerdidaPorAdicion\ (j,x)\big)\ j\ \ni\ s_x \quad (15)$$

Sea s_x el conjunto de instalaciones seleccionadas a ubicar, es decir, que su estado en la solución es 1; δ_i es la distancia entre la demanda i y la instalación más cercana j que pertenece a s_x; dij La distancia entre una demanda i y la instalación j que pertenece a s_x.

- Segundo caso. Cuando el número de instalaciones es mayor a P, para lo cual la ecuación (16) evalúa el conjunto de puntos seleccionados como instalaciones y determina para cada punto cuando aumenta la función objetivo al ser este eliminado del conjunto de instalaciones. La ecuación (17) selecciona la instalación que genere el mayor aumento de la función objetivo y procederá a quitarlo de la lista de instalaciones de la solución.

$$GananciaPorEliminacion\ (j.x) = \sum_{i\in I\ y\ j=argmin_{t\in s_x}\{d_{it}\}} (min\{d_{it}, t \in\ s_x \setminus \{j\}\} - \delta_i) \quad (16)$$

$$k = \ Argmax\ \big(GanaciaPorEliminacion\ (j,x)\big)\ j\ \ni\ s_x \quad (17)$$

Sea t el conjunto de instalaciones que quedan después de quitar una instalación j de s_x, dit es la distancia entre una demanda i y la instalación más cercana t que pertenece a s_x.

Tomando como base lo anterior se realizan cambios para adaptarlo al problema de *P*-mediana, que busca ubicar instalaciones lo más cerca posible de los clientes para minimizar la función objetivo. Estos cambios se realizaron al agregar la instalación (ver ecuación (18)), seleccionando el vértice j con mayor valor de perdida de la función objetivo; y al eliminar la instalación (ver ecuación (19)) seleccionando el punto j que pertenece al conjunto de vértices con menor aumento de la función. Luego se repiten estos procedimientos hasta que el número de instalaciones en la solución sea igual a P.

$$k = \ DeterminarPosArgmax\ \big(PerdidaPorAdicion\ (j,x)\big)\ j\ \ni\ s_x \quad (18)$$

$$k = DeterminarPosArgmin\left(GanaciaPorEliminacion\,(j,x)\right) j \ni s_x \qquad (19)$$

Como se muestra en la Figura 7, el operador de reparación inicia determinando las menores distancias entre las instalaciones que han sido ubicadas en el espacio geográfico, y los puntos de demanda (línea 3), si el número de instalaciones ubicadas en el espacio geográfico es menor a la restricción P, se determina entre los puntos de demanda (vértice) que no tienen asignada una instalación, el punto que genera una mayor disminución en el valor de la función objetivo (línea 5) para ser activado como instalación (línea 6). Por otro lado, si el número de instalaciones ubicadas es mayor a la restricción P, se establece que instalación al ser eliminada genera el mínimo aumento en el valor de la función objetivo (línea 11), para ser desactivada como una instalación (línea 13). En ambos casos, la lista de menores distancias es actualizada (línea 7 y 14).

```
1   INICIO
2         pMedianas = MiProblema.Pmedianas
3         menoresDistancias = DeterminarMenoresDistancias(posInstalaciones)
4         MIENTRAS(PosInstalaciones.contar  < pMedianas) HACER
5                pos = DeterminarPosArgMax(menoresDistancias, Vertices)
6                Activar(Pos)
7                menoresDistancias=ActualizarMenoresDistanciasAgregacion(
8                            MyAlgorithm, pos, menoresDistancias)
9         FINMIENTRAS
10        MIENTRAS (PosInstalaciones.contar > pMedianas) HACER
11               pos = DeterminarPosArgMin(menoresDistancias,
12                           PosInstalaciones, MyAlgorithm)
13               InActivar(pos)
14               menoresDistancias=ActualizarMenoresDistanciasEliminacion(
15                           MyAlgorithm, pos, menoresDistancias, PosInstalaciones)
16        FINMIENTRAS
17  FIN
```

Figura 7. Reparador Con Conocimiento

3.1.5.2 Reparador aleatorio

En este reparador (ver Figura 8) cuando hay menos instalaciones ubicadas en la solución (línea 3), se ubica una instalación de forma aleatoria del conjunto de puntos de demanda en donde esta puede ser ubicada (línea 4) y se agrega a la solución. De igual forma, si sobran instalaciones, el reparador toma aleatoriamente una instalación que ha sido ubicada para ser eliminada del conjunto de instalaciones (línea 7). Luego, se repiten estos procedimientos hasta que el número de instalaciones en la solución sea igual a *P*

```
1   INICIO
2         pMedianas = MyAlgorithmo.MyProblema.PMedianas
3         MIENTRAS (Contar(PosInstalaciones) < pMedianas) HACER
4                Activar(Utils.VerticeValidacion(Vertices,MyAlgorithm, 1));
5         FIN MIENTRAS
6         MIENTRAS (Contar(PosInstalaciones) > pMedianas) HACER
7                InActivar(Utils.VerticeValidation(Vertices, MyAlgorithm, 0));
8         FIN MIENTRAS
9   FIN
```

Figura 8. Reparador Aleatorio

3.1.6 Resultados Preliminares

A partir de estos parámetros iniciales de la Tabla 3, (ver sección 3.1.2) y de las adaptaciones realizadas, se obtienen los resultados en la Tabla 5, utilizando el operador de inicialización aleatorio con conocimiento (ver sección 3.1.4.2), y el operador de reparación con conocimiento (ver sección 3.1.5.1)

Tabla 5. Resultados parámetros PAR=0.365 HMCR=1-(10/NP) HMS=30

Conjunto de datos		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed1	5819	5819.00	5820.03	0	0.0177	0	0.0103
pmed2	4093	4093.40	4095.20	0.0097	0.0537	0.0040	0.0220
pmed3	4250	4250.00	4251.73	0	0.0407	0	0.0173
pmed4	3034	3036.73	3056.37	0.0900	0.7372	0.0273	0.2236
pmed5	1355	1371.50	1409.73	1.2177	4.0393	0.1650	0.5473
pmed6	7824	7824.00	7824.00	0	0	0	0
pmed7	5631	5631.00	5632.23	0	0.0219	0	0.0123
pmed8	4445	4445.00	4468.47	0	0.5279	0	0.2346
pmed9	2734	2750.87	2840.53	0.6169	3.8966	0.1686	1.0653
pmed10	1255	1309.37	1434.13	4.3320	14.2735	0.5436	1.7913
pmed11	7696	7696.00	7696.70	0	0.0090	0	0.0070
pmed12	6634	6634.00	6638.60	0	0.0693	0	0.0460
pmed13	4374	4374.10	4437.90	0.0022	1.4609	0.0010	0.6390
pmed14	2968	2982.70	3157.33	0.4952	6.3791	0.1470	1.8933
pmed15	1729	1802.83	1971.63	4.2702	14.0331	0.7383	2.4263
pmed16	8162	8162.00	8162.17	0	0.0020	0	0.0016
pmed17	6999	6999.00	7004.57	0	0.0795	0	0.0556
pmed18	4809	4811.73	4914.90	0.0568	2.2021	0.0273	1.0590
pmed19	2845	2868.70	3060.63	0.8330	7.5793	0.2370	2.1563
pmed20	1789	1891.87	2122.57	5.7499	18.6454	1.0286	3.3356
pmed21	9138	9138.00	9138.00	0	0	0	0
pmed22	8579	8579.00	8580.73	0	0.0202	0	0.0173
pmed23	4619	4623.87	4779.00	0.1053	3.4639	0.0486	1.6000
pmed24	2961	2988.07	3218.67	0.9141	8.7020	0.2706	2.5766
pmed25	1828	1936.63	2162.47	5.9427	18.2968	1.0863	3.3446
pmed26	9917	9917.00	9918.70	0	0.0171	0	0.0170
pmed27	8307	8307.10	8312.77	0.0012	0.0694	0.0010	0.0576
pmed28	4498	4506.07	4659.13	0.1793	3.5823	0.0806	1.6113
pmed29	3033	3057.93	3310.40	0.8220	9.1460	0.2493	2.7740
pmed30	1989	2107.53	2363.77	5.9594	18.8419	1.1853	3.7476
			Promedio	1.0533	4.5403	0.2003	1.0430

Además, en la Tabla 6 se presentan los resultados obtenidos con los mismos parámetros iniciales mencionados en la sección 3.1.2, con el operador de inicialización aleatorio controlado (ver sección 3.1.4.1), y el operador de reparación aleatorio (ver sección 3.1.5.2).

Tabla 6. Resultados parámetros PAR=0.5 HMCR=0.95 HMS=25

Conjunto de datos		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed1	5819	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed2	4093	5976.2333	5935.4333	2.7020	2.0009	1.5723	1.1643
pmed3	4250	4585.0333	4459.6666	12.0213	8.9583	4.9203	3.6666
pmed4	3034	4744.9000	4662.0000	11.6447	9.6941	4.9490	4.1200
pmed5	1355	3629.3666	3595.5333	19.6231	18.5080	5.9536	5.6153
pmed6	7824	1792.0000	1808.2000	32.2509	33.4460	4.3700	4.5320
pmed7	5631	8095.3333	8043.4666	3.4679	2.8050	2.7133	2.1946
pmed8	4445	6164.1666	6129.0333	9.4684	8.8444	5.3316	4.9803
pmed9	2734	5424.6666	5266.3666	22.0397	18.4780	9.7966	8.2136
pmed10	1255	3586.7333	3536.2333	31.1899	29.3420	8.5273	8.0223
pmed11	7696	1839.9000	1859.8333	46.6055	48.1930	5.8490	6.0483
pmed12	6634	7955.4000	7981.7000	3.3705	3.7123	2.5940	2.8570
pmed13	4374	7435.8666	7282.6666	12.0872	9.7779	8.0186	6.4866
pmed14	2968	5379.1333	5240.7333	22.9797	19.8150	10.0513	8.6673
pmed15	1729	3938.7333	3878.9000	32.7066	30.6900	9.7073	9.1090
pmed16	8162	2521.0000	2514.8000	45.8068	45.4480	7.9200	7.8580
pmed17	6999	8468.1333	8503.2666	3.7507	4.1811	3.0613	3.4126
pmed18	4809	7771.533	7731.8000	11.0377	10.4700	7.7253	7.3280
pmed19	2845	5913.3333	5819.2333	22.9638	21.0070	11.0433	10.1023
pmed20	1789	3831.7666	3797.4666	34.6842	33.4780	9.8676	9.5246
pmed21	9138	2713.0333	2739.6333	51.6508	53.1370	9.2403	9.5063
pmed22	8579	9567.9333	9730.2666	4.7048	6.4813	4.2993	5.9226
pmed23	4619	9408.8333	9421.4000	9.6728	9.8193	8.2983	8.4240
pmed24	2961	5798.5333	5711.1333	25.5365	23.6440	11.7953	10.9213
pmed25	1828	4020.0333	4000.7333	35.7660	35.1140	10.5903	10.3973
pmed26	9917	2790.2000	2808.9000	52.6367	53.6590	9.6220	9.8090
pmed27	8307	1026.7666	10461.3666	3.5370	5.4892	3.5076	5,4436
pmed28	4498	9155.7333	9105.5333	10.2170	9.6127	8.4873	7.9853
pmed29	3033	5718.7000	5618.6333	27.1387	24.9140	12.2070	11.2063
pmed30	1989	4151.4333	4119.1000	36.8754	35.8040	11.1843	10.8610
			Promedio	22.9880	22.2319	7.4482	7.1571

En la experimentación se encontraron mejores resultados haciendo uso de los operadores de inicialización con conocimiento y del reparador con conocimiento (ver Tabla 5), esto es debido a que estos operadores hacen una búsqueda exhaustiva. Por esto, con esta configuración de los algoritmos se inicia el afinamiento de parámetros.

3.1.7 Afinamiento de parámetros

Debido a la complejidad del problema, se investigó en la literatura sobre posibles valores de estos parámetros. Sin embargo, no se encontró una aproximación para el problema de ubicación de instalaciones *P*_mediana no capacitado, por lo tanto, se partió de los valores

preliminares de los parámetros obtenidos en la Tabla 3 (sección 3.1.2) y se realizó una exploración de valores cercanos a estos indicados en el problema de la mochila (ver Tabla 7).

Además, se agregaron los parámetros: (1) límite inferior ($LimInf$), el cual toma un valor del 25% para el algoritmo SBHS y del 50% para HSOS de las medianas a ubicar en el espacio, teniendo un mínimo en el límite de instalaciones presentes en una solución durante la evolución del algoritmo, para no obtener soluciones inviables (cuando no se ha ubicado una instalación en la solución); y (2) límite superior ($LimSup$), que está presente en el algoritmo HSOS, este parámetro toma un valor del 50% extra de medianas a ubicar, restringiendo el número máximo de posiciones que se activarán como instalaciones en una solución, para disminuir el tiempo de reparación dado que en la evolución se puede generar soluciones con muchas instalaciones activadas, generando un mayor tiempo de procesamiento en la reparación. En el caso de SBHS no fue necesario este parámetro, debido a que no genera soluciones con muchas instalaciones activadas.

Tabla 7. Ajuste de parámetros

Parámetro		Valor	
SBHS	HMS HMCR LimInf	25 0.95 NumVertices*0.25	35 (1-(1/NumVertices)) NumVertices*0.25
HSOS	HMS PAR HMCR LimInf LimSup	25 0.5 0.95 NumVertices*0.5 NumVertices*1.5	35 0.365 (1-(1/NumVertices)) NumVertices*0.5 NumVertices*1.5

Partiendo del conjunto de valores de los parámetros de la Tabla 7, de la primera columna se obtuvieron los resultados que se presentan en la Tabla 8, y de la segunda columna los que se muestran en la Tabla 9.

Tabla 8. Resultados parámetros PAR=0.5 HMCR=0.95 HMS=25

Problema		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Conjunto de datos	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed1	5819	5819	5819	0	0	0	0
pmed2	4093	4093	4095	0	0.0488	0	0.0200
pmed3	4250	4250	4250	0	0	0	0
pmed4	3034	3034.67	3052.63	0.0219	0.6141	0.0066	0.1863
pmed5	1355	1368.17	1413.40	0.9717	4.3099	0.1316	0.5840
pmed6	7824	7824	7824	0	0	0	0
pmed7	5631	5632.40	5631.27	0.0248	0.0047	0.0140	0.0026
pmed8	4445	4445	4468.70	0	0.5331	0	0.2370
pmed9	2734	2748.57	2828.37	0.5327	3.4515	0.1456	0.9436

Problema		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Conjunto de datos	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed10	1255	1305.67	1427.03	4.0371	13.7078	0.5066	1.7203
pmed11	7696	7696	7696	0	0	0	0
pmed12	6634	6634	6634	0	0	0	0
pmed13	4374	4374.33	4443.50	0.0076	1.5889	0.0033	0.6950
pmed14	2968	2980.97	3142.77	0.4368	5.8883	0.12966	1.7476
pmed15	1729	1804.30	1964.03	4.3551	13.5935	0.7530	2.3503
pmed16	8162	8162	8162	0	0	0	0
pmed17	6999	6999	7000.27	0	0.0180	0	0.0126
pmed18	4809	4811.30	4911.37	0.0478	2.1286	0.0230	1.0236
pmed19	2845	2864.63	3041.07	0.6900	6.8916	0.1963	1.9606
pmed20	1789	1875.73	2100.97	4.8481	17.4380	0.8673	3.1196
pmed21	9138	9138	9138	0	0	0	0
pmed22	8579	8579	8579	0	0	0	0
pmed23	4619	4621.67	4759.27	0.0577	3.0367	0.0266	1.4026
pmed24	2961	2984.67	3198.83	0.7992	8.0321	0.2366	2.3783
pmed25	1828	1927.13	2166.43	5.4230	18.5138	0.9913	3.3843
pmed26	9917	9917	9917.47	0	0.0047	0	0.0046
pmed27	8307	8307.10	8307.10	0.0012	0.0012	0.0010	0.0010
pmed28	4498	4504.72	4650.03	0.1494	3.3800	0.0672	1.5203
pmed29	3033	3050.83	3300.27	0.5879	8.8119	0.1783	2.6726
pmed30	1989	2100.90	2364	5.6259	18.8536	1.1190	3.7500
			Promedio	**0.9539**	**4.3616**	**0.1799**	**0.9905**

Tabla 9. Resultados parámetros PAR=0.365 HMCR = (1-(1/NumVertices)) HMS=35.

Problema		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Conjunto de datos	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed1	5819	5819	5820.03	0	0.0177	0	0.0103
pmed2	4093	4093.40	4095.93	0.0097	0.0716	0.0040	0.0293
pmed3	4250	4250	4251.73	0	0.0407	0	0.0173
pmed4	3034	3036.73	3056.37	0.0900	0.7372	0.0273	0.2237
pmed5	1355	1371.50	1409.73	1.2177	4.0393	0.1650	0.5473
pmed6	7824	7824	7824	0	0	0	0
pmed7	5631	5631	5632.23	0	0.0219	0	0.0123
pmed8	4445	4445	4468.47	0	0.5279	0	0.2347
pmed9	2734	2750.86	2840.53	0.6169	3.8966	0.1687	1.0653
pmed10	1255	1309.36	1434.13	4.3320	14.2735	0.5437	1.7913
pmed11	7696	7696	7696.70	0	0.009	0	0.0070
pmed12	6634	6634	6638.60	0	0.0693	0	0.0460
pmed13	4374	4374.10	4436.40	0.0022	1.4266	0.001	0.6240
pmed14	2968	2982.70	3157.33	0.4952	6.3791	0.147	1.8933
pmed15	1729	1802.83	1971.63	4.2702	14.0331	0.73833	2.4263
pmed16	8162	8162	8162.17	0	0.0020	0	0.0017

Problema		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Conjunto de datos	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed17	6999	6999	7004.57	0	0.0795	0	0.0557
pmed18	4809	4811.73	4914.90	0.0568	2.2021	0.0273	1.0590
pmed19	2845	2868.70	3060.63	0.8330	7.5793	0.2370	2.1563
pmed20	1789	1891.86	2122.57	5.7499	18.6454	1.0287	3.3357
pmed21	9138	9138	9138	0	0	0	0
pmed22	8579	8579	8580.77	0	0.0205	0	0.0177
pmed23	4619	4623.93	4779	0.1068	3.4639	0.0493	1.6000
pmed24	2961	2988.07	3218.67	0.9141	8.7020	0.2707	2.5767
pmed25	1828	1936.63	2162.47	5.9427	18.2968	1.08633	3.3447
pmed26	9917	9917	9918.70	0	0.0171	0	0.0170
pmed27	8307	8307.10	8312.77	0.0012	0.0694	0.0010	0.0577
pmed28	4498	4509.37	4659.13	0.2527	3.5823	0.1137	1.6113
pmed29	3033	3057.93	3310.40	0.8220	9.1460	0.2493	2.7740
pmed30	1989	2110.5	2363.77	6.1085	18.8419	1.2150	3.7477
			Promedio	1.0607	4.5397	0.2024	1.0428

Obteniendo los mejores resultados a partir de la Tabla 8, con los parámetros que se muestran en la Tabla 10.

Tabla 10. Parámetros finales ciclo I.

Parámetro		Valor
SBHS	HMS HMCR LimInf	25 0.95 NumVertices*0.25
HSOS	HMS PAR HMCR LimInf LimSup	25 0.5 0.95 NumVertices*0.50 NumVertices*1.50

Para el algoritmo HSOS, el parámetro PAR permitió definir la probabilidad de modificar de igual manera el estado en una dimensión de la armonía (1, en caso que se desea ubicar una instalación, y 0 al momento de quitar esta), además que al disminuir el tamaño de la población (HMS) en ambos algoritmos, permitió obtener en las nuevas armonías una mayor calidad al momento de ser evaluadas en el proceso de evolución, y por último al disminuir el valor de $HMCR$ para ambos algoritmos, permitió la inserción de diferentes armonías de forma aleatoria en la población, logrando una mayor exploración debido al número de vértices que tiene cada problema.

3.2 CICLO II: Adición búsqueda local a los algoritmos adaptados

3.2.1 Búsqueda local

Una vez finalizado el ciclo I, se estudió e identificó que búsqueda local sería la más apropiada para el problema FLP. Teniendo en cuenta el tamaño de los conjuntos de datos y el tiempo de ejecución de los algoritmos, se determinó que el algoritmo de búsqueda local más adecuado era el ascenso de la colina, en el cual se realiza un movimiento (cambio) a la solución actual por un determinado número de veces.

Debido a que en el problema de *P*-mediana, la forma de realizar un cambio a una solución es apagando una instalación (estado en 0) y prendiendo un punto de demanda (estado en 1), se definieron dos formas de hacer este movimiento aplicando activación y cierre de instalaciones de forma aleatoria y con conocimiento.

3.2.1.1 Activación de una instalación de forma aleatoria

Para el desarrollo del operador de activación de una instalación de forma aleatoria (Ver Figura 9), se selecciona un punto de demanda en el espacio geográfico de forma aleatoria (Línea 4), es decir que su valor sea 0, y además que no sea una instalación (Línea 5), luego retorna su posición (Línea 6) para cambiar su estado a 1 dentro de la solución.

```
1  INICIO seleccionarInstalacion()
2    pos = -1
3    HACER
4      pos = validarvertice(Solution.vertices, Solution.MyAlgoritmo, 0)
5    MIENTRAS verticesnoposibles.contengan(pos);
6    retornar pos
7  FIN
```

Figura 9. Activación de una instalación de forma aleatoria

3.2.1.2 Activación de una instalación con conocimiento

Para el desarrollo del operador de activación de una instalación con conocimiento (ver Figura 10), el algoritmo toma cada uno de los puntos de demanda que no sean instalaciones y que no pertenezcan a lista de vecinos inviables para ser activados (línea 4), es decir, las ubicaciones que no han sido probadas y calcula su distancia de acuerdo con las distancias de Floyd con la instalación que se pretende activar (línea 5) y con base en los valores calculados y guardados para los puntos de demanda (línea 6), se selecciona aquel que presente el mínimo valor de distancia (línea 9) para ser activada.

```
1  INICIO PrenderVecinoMascerano (vecinosinviables, instalacionApagar)
2    Distancias = Lista<índice, valor>;
3    DESDE i = 0 HACER i < NumVertices
4      Si PosInstalaciones y vecinosinviables nocontienen(i)
5        Dist = DistanciasFloyd[instalacionapagada][i]
6        Distancias.añadir(i, dist)
7      FINSI
8    FIN DESDE
9    return pos = min(Distancias)
10 FIN
```

Figura 10. Activacion de una instalación con conocimiento

3.2.1.3 Cierre de una instalación de forma aleatoria

Para el cierre de una instalación de forma aleatoria (ver Figura 11), selecciona una instalación que ha sido ubicada en el espacio geográfico de forma aleatoria (Línea 4) y que no esté en la lista de vértices no posibles para ser apagados (Línea 5), luego retorna su posición (Línea 6) para cambiar su estado a 0 dentro de la solución.

```
1  INICIO seleccionarInstalacion()
2    pos = -1
3    HACER
4      pos = validarvertice(vertices, MyAlgoritmo, 1)
5    MIENTRAS verticesnoposibles.contengan(pos);
6    retornar pos
7  FIN
```

Figura 11. Cierre de una instalación de forma aleatoria

3.2.1.4 Cierre de una instalación con conocimiento

Para seleccionar con conocimiento la instalación a cerrar (ver Figura 12) el algoritmo toma cada una de las instalaciones que están activadas en la solución (línea 4) y calcula la suma de distancias hacia todos los puntos de demanda de acuerdo con las distancias de Floyd (línea 7), con base en los valores calculados y guardados para cada instalación (línea 9), seleccionando aquella que presente el máximo valor (líneas 13 ,14).

```
1  INICIO seleccionarInstalacionCerrar(inviablesCerrar, Solucion)
2    distancias = lista<índice, valor>();
3    DESDE t = 0 HASTA contar(InstalacionesSolucion)
4      SI vecinos_no_viables NO_contiene(InstalacionesSolucion[t]) ENTONCES
5        Sum = 0
6        DESDE i = 0 HASTA Numvertices_Problema
7          Sum = Sum+DistanciasFloyd(Solucion[i][InstalacionesSolucion[t]])
8        FIN DESDE
9        Añadir(distancias, <InstalacionesSolucion[t], sum>)
10     FIN SI
11   FIN DESDE
13   mas = Máximo(distancias)
14   retornar posK = encontrar(valor(distancias, mas))
15 FIN
```

Figura 12. Cierre de una instalación con conocimiento

3.2.2 Parámetros iniciales

Para la ejecución del algoritmo de búsqueda local, se incluyeron nuevos parámetros asociados a la búsqueda local: (1) $NumVecinos$, que indica el número de movimientos que se realizaran a la solución; (2) $ProbabilidadAplicarLS$, probabilidad de aplicar la búsqueda local implementada en estos algoritmos.

Para definir los valores iniciales de estos parámetros se tiene en cuenta que: el número de evaluaciones de la función objetivo es limitado y que el algoritmo HSOS modifica en cada iteración todas las soluciones de la población actual, por lo tanto, la probabilidad de que se aplique la búsqueda local y el número de vecinos generados para cada solución debe ser pequeño. Mientras que el algoritmo SBHS genera una única solución en cada iteración, por lo cual, la probabilidad de aplicar la búsqueda local puede ser alta y el número de vecinos

generados se puede incrementar un poco. Estos parámetros iniciales se muestran en la Tabla 11.

Tabla 11. Parámetros iniciales para los algoritmos con LS

Parámetro		Valor
SBHS-LS	NumVecinos ProbabilidadAplicarLS	5 100
HSOS-LS	NumVecinos ProbabilidadAplicarLS	3 30

3.2.3 Resultados parciales

Debido a la complejidad del problema y el tiempo utilizado para el procesamiento del conjunto de datos usado en el Ciclo I, se optó para el Ciclo II, realizar un agrupamiento inicial de instancias de acuerdo con el número de puntos de demanda y las medianas, para determinar qué conjunto de datos ejecutar y así reducir los tiempos de ejecución durante la experimentación. Para definir estos grupos, se tomaron aquellas instancias que tenían una cantidad diferente de medianas para distintos vértices en los puntos de demanda.

La experimentación se realizó haciendo algunas combinaciones de las diferentes activaciones y cierres de instalaciones mencionados en la sección 3.2.1, para realizar el movimiento en la búsqueda local.

En la Tabla 12 se describe los resultados obtenidos al hacer uso de los operadores de activación de una instalación con conocimiento y de cierre de una instalación con conocimiento.

Tabla 12. Prender con conocimiento, apagar por conocimiento

Problema				Promedio		RPE	
Conjunto de datos	Vértices	Mediana	Óptimo	SBHS	HSOS	SBHS	HSOS
pmed1	100	5	5819	5819	5819	0	0
pmed4	100	20	3034	3046.1333	3046.1333	3.0608	0.3999
pmed6	200	5	7824	7824	7824	0	0
pmed8	200	20	4445	4446.4000	4446.4000	1.7975	0.0314
pmed14	300	60	2968	3008.9000	3008.9000	7.8099	1.3780
pmed19	400	80	2845	2898.3667	2898.3667	8.0023	1.8758
pmed23	500	50	4619	4642.7333	4642.7333	4.1271	0.5138
pmed26	600	5	9917	9917	9917	0.0047	0
					Promedio RPE	3.1003	0.5249

En la Tabla 13 se describe los resultados obtenidos al hacer uso de los operadores de activación de una instalación de forma aleatoria y cierre de una instalación de forma aleatoria.

Tabla 13. Prender aleatorio, apagar aleatorio.

Problema				Promedio		RPE	
Conjunto de datos	Vértices	mediana	Óptimo	SBHS	HSOS	SBHS	HSOS
pmed1	100	5	5819	5819.0667	5819	0.0011	0
pmed4	100	20	3034	3126.5667	3044.6333	3.0509	0.3504
pmed6	200	5	7824	7824	7824	0	0
pmed8	200	20	4445	4531.2667	4446.6000	1.9407	0.0359
pmed14	300	60	2968	3189	3005.6333	7.4460	1.2679
pmed19	400	80	2845	3070.7667	2894.9333	7.9355	1.7551
pmed23	500	50	4619	4809.7667	4640.1000	4.1300	0.4568
pmed26	600	5	9917	9917.8333	9917	0.0084	0
					Promedio RPE	3.0641	0.4833

Y por último en la Tabla 14, se describe los resultados obtenidos al hacer uso de los operadores de activación de una instalación de forma aleatoria y cierre de una instalación con conocimiento.

Tabla 14. Prender Aleatorio, apagar conocimiento.

Problema				**Promedio**		**RPE**	
Conjunto de datos	Vértices	mediana	Óptimo	SBHS	HSOS	SBHS	HSOS
pmed1	100	5	5819	5819	5819	0	0
pmed4	100	20	3034	3108.5667	3043.7667	2.4577	0.3219
pmed6	200	5	7824	7824	7824	0	0
pmed8	200	20	4445	4529.7000	4447.8333	1.9055	0.0637
pmed14	300	60	2968	3199.3000	3009.9667	7.7931	1.4139
pmed19	400	80	2845	3070.9667	2896.8667	7.9425	1.8230
pmed23	500	50	4619	4787.8000	4642.5333	3.6544	0.5094
pmed26	600	5	9917	9917.4667	9917	0.0047	0
					Promedio RPE	2.9697	0.5165

Con estos resultados, se hace una comparación de los movimientos usados (ver Tabla 15), por medio de la medida de función de error porcentual, que permite conocer el error relativo al hacer uso de estos operadores en el algoritmo de búsqueda local.

Tabla 15. Comparación movimiento para búsqueda local.

	PROMEDIO RPE		
TIPO PRUEBA	SBHS	HSOS	TOTAL
PrenderConocimiento_ApagarAleatorio	3.1003	0.5249	3.6252
PrenderAleatorio_ApagarAleatorio	3.0641	0.4833	3.5473
PrenderAleatorio_ApagarConocimiento	**2.9697**	**0.5165**	**3.4862**

Como se observa en la Tabla 15, se obtiene que activar una instalación de forma aleatoria y cerrar con conocimiento obtiene mejores resultados, permitiendo diversidad al seleccionar una instalación del espacio geográfico, y en el cierre busca disminuir el valor de la función objetivo entre los valores que han sido ubicados.

Por otra parte, la búsqueda local se puede implementar tanto en la evolución de los algoritmos adaptados, como en la inicialización de la población, por ende, se hace una experimentación para observar si es una buena opción implementar la búsqueda local sobre las soluciones de la población inicial, tal como se muestra en la Tabla 16.

Tabla 16. Comparación aplicar búsqueda local.

		CON LS EN POBLACION INICIAL		SIN LS EN POBLACION INICIAL	
Problema		RPE		RPE	
Conjunto de datos	Óptimo	SBHS	HSOS	SBHS	HSOS
pmed1	5819	0.00	0.00	0	0
pmed4	3034	2.97	0.52	2.4577	0.3219
pmed6	7824	0.00	0.00	0	0
pmed8	4445	2.11	0.07	1.9055	0.0637
pmed14	2968	7.74	1.82	7.7931	1.4139
pmed19	2845	8.41	2.32	7.9425	1.8230
pmed23	4619	4.00	0.74	3.6544	0.5094
pmed26	9917	0.00	0.00	0.0047	0
Promedio		**3.15**	**0.68**	**2.9697**	**0.5165**

Como se observa en la Tabla 16, los resultados no mejoran al aplicar el algoritmo de búsqueda local en el operador de inicialización, esto puede ocurrir debido a que los individuos pueden quedarse en óptimos locales y de esta forma no se logra explorar otros puntos del espacio de búsqueda.

3.2.4 Afinamiento final de parámetros

Para este afinamiento de parámetros, se tuvo en cuenta los operadores usados en el afinamiento de parámetros para el ciclo I, pero debido a que la desviación estándar y la función de RPE se estaba incrementando con respecto a los resultados obtenidos en el ciclo I, se disminuyó la probabilidad de aplicar el algoritmo de búsqueda local y se incrementó el número de vecinos generados a la solución actual, tomando valores cercanos a los encontrados en los resultados parciales, tal como se muestra en la Tabla 17.

Tabla 17. Parámetros Finales HSOS-SBHS.

Parámetro		Valor	
SBHS-LS	NumVecinos ProbabilidadAplicarLS	10 20%	15 10%
HSOS-LS	NumVecinos ProbabilidadAplicarLS	10 15%	15 10%

Partiendo del conjunto de valores de los parámetros de la Tabla 17, de la primera columna se obtuvieron los resultados que se presentan en la Tabla 18, y de la segunda columna los que se muestran en la Tabla 19.

Tabla 18. Resultados Numvecinos=10; Probabilidad=20% SBHS; Probabilidad=15% HSOS.

Conjunto de datos		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed1	5819	5819	5819	0	0	0	0
pmed2	4093	4093	4093.8000	0	0.0196	0	0.0080
pmed3	4250	4250	4250.8667	0	0.0204	0	0.0087
pmed4	3034	3045.40000	3088.3667	0.3757	1.7919	0.1140	0.5437
pmed5	1355	1392.06667	1437.3333	2.7355	6.0763	0.3707	0.8233
pmed6	7824	7824	7824	0	0	0	0
pmed7	5631	5631.9333	5634.4333	0.0166	0.0610	0.0093	0.0343
pmed8	4445	4447.4667	4496.7333	0.0555	1.1639	0.0247	0.5173
pmed9	2734	2774.7667	2862.7333	1.4911	4.7086	0.4077	1.2873
pmed10	1255	1348.1000	1444.7667	7.4183	15.1208	0.9310	1.8977
pmed11	7696	7696	7696	0	0	0	0
pmed12	6634	6634.7667	6635.6333	0.0116	0.0246	0.0077	0.0163
pmed13	4374	4393.0333	4468.7333	0.4351	2.1658	0.1903	0.9473
pmed14	2968	3023.5333	3177.8333	1.8711	7.0699	0.5553	2.0983
pmed15	1729	1850.1333	1985.4000	7.0060	14.8294	1.2113	2.5640
pmed16	8162	8162	8162.7000	0	0.0086	0	0.0070
pmed17	6999	6999.2667	6999.8000	0.0038	0.0114	0.0027	0.0080
pmed18	4809	4834.5667	4935.4333	0.5316	2.6291	0.2557	1.2643
pmed19	2845	2909.2333	3059.9667	2.2578	7.5559	0.6423	2.1497
pmed20	1789	1939.7333	2132.0667	8.4256	19.1764	1.5073	3.4307
pmed21	9138	9138	9138	0	0	0	0
pmed22	8579	8579.1667	8579.3333	0.0019	0.0039	0.0017	0.0033
pmed23	4619	4655.5667	4795.2333	0.7917	3.8154	0.3657	1.7623
pmed24	2961	3035.9000	3224.4667	2.5296	8.8979	0.749	2.6347
pmed25	1828	1993.8333	2178.7000	9.0718	19.1849	1.65833	3.5070
pmed26	9917	9917	9917.2333	0	0.0024	0	0.0023
pmed27	8307	8307.8000	8309.1000	0.0096	0.0253	0.0080	0.0210
pmed28	4498	4529.7333	4681.8667	0.7055	4.0877	0.3173	1.8387
pmed29	3033	3101.8000	3327.6667	2.2684	9.7154	0.6880	2.9467
pmed30	1989	2167.5000	2380.3333	8.9744	19.6749	1.7850	3.9133
			Promedio	1.8996	4.9280	0.3934	1.1412

Tabla 19. Resultados Numvecinos=15; Probabilidad=10%.

Conjunto de datos		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed1	5819	5819.0000	5819.0000	0	0	0	0
pmed2	4093	4093.4000	4093.5000	0.0097	0.0382	0.004	0.0050
pmed3	4250	4250.0000	4250.0000	0	0.0447	0	0

Conjunto de datos		Promedio		FmedianRPE		Desviación estándar	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed4	3034	3050.9666	3045.1667	0.5592	1.7765	0.1697	0.1117
pmed5	1355	1394.0000	1385.9333	2.8782	6.6199	0.3900	0.3093
pmed6	7824	7824.0000	7824.0000	0	0	0	0
pmed7	5631	5631.7333	5631.4667	0.013	0.0674	0.0073	0.0046
pmed8	4445	4449.5333	4447.0000	0.1019	1.4165	0.0453	0.0200
pmed9	2734	2778.4667	2766.6000	1.6264	5.0865	0.4447	0.3260
pmed10	1255	1354.6333	1330.6000	7.9389	15.6148	0.9963	0.7560
pmed11	7696	7696.0000	7696.0000	0	0	0	0
pmed12	6634	6637.4000	6634.5667	0.0512	0.0678	0.0340	0.0056
pmed13	4374	4399.8000	4382.1667	0.5898	2.4234	0.2580	0.0817
pmed14	2968	3023.4667	3015.6333	1.8688	7.2708	0.5546	0.4763
pmed15	1729	1851.0333	1837.0333	7.0580	15.3634	1.2203	1.0803
pmed16	8162	8162.0000	8162.0000	0	0	0	0
pmed17	6999	6999.3333	6999.0000	0.0047	0.0447	0.0033	0
pmed18	4809	4837.7667	4822.8667	0.5981	2.8162	0.2876	0.1387
pmed19	2845	2910.6000	2893.6000	2.3057	7.5383	0.656	0.486
pmed20	1789	1941.9667	1924.5333	8.5504	19.0702	1.5297	1.3553
pmed21	9138	9138.0000	9138.0000	0	0	0	0
pmed22	8579	8579.4000	8579.1333	0.0046	0.0528	0.0040	0.0013
pmed23	4619	4657.8667	4640.4667	0.8414	3.8918	0.3887	0.2146
pmed24	2961	3024.3000	3020.5333	2.1377	8.9237	0.6330	0.5953
pmed25	1828	1988.9333	1971.9667	8.8037	19.8887	1.6093	1.4396
pmed26	9917	9917.0000	9917.0000	0	0	0	0
pmed27	8307	8307.8333	8308.0000	0.01	0.0497	0.0083	0.0100
pmed28	4498	4529.3667	4516.8667	0.6973	3.9424	0.3137	0.1886
pmed29	3033	3103.8000	3088.4667	2.3343	9.5889	0.708	0.5547
pmed30	1989	2169.8000	2152.9333	9.0899	19.348	1.808	1.6393
			Promedio	1.9358	5.0315	0.4025	0.3267

Por último, como con estos parámetros no se obtenían mejores resultados con respecto al Ciclo I, se optó por dar un menor número de vecinos y una mayor probabilidad a la explotación por parte del algoritmo de búsqueda local. En la Tabla 20, se muestran los resultados obtenidos al aplicar el valor de 5 en el parámetro de $NumVecinos$ y una probabilidad de 100%.

Tabla 20. Resultados NumVecinos=5; Probabilidad=100%.

Conjunto de datos		Promedio		RPE		Desviación estándar	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed1	5819	5819.0000	5819.0000	0	0	0	0
pmed2	4093	4093.5000	4099.4666	0.0122	0.1579	0.0050	0.0646
pmed3	4250	4250.0000	4258.3333	0	0.196	0	0.0833
pmed4	3034	3045.1666	3108.5666	0.3680	2.457	0.1116	0.7456
pmed5	1355	1385.9333	1464.8666	2.2829	8.108	0.3093	1.0986

Conjunto de datos		**Promedio**		**RPE**		**Desviación estándar**	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed6	7824	7824.0000	7824.0000	0	0	0	0
pmed7	5631	5631.4666	5639.9333	0.0082	0.158	0.0046	0.0893
pmed8	4445	4447.0000	4529.7000	0.0449	1.905	0.0200	0.8470
pmed9	2734	2766.6000	2894.1000	1.1923	5.855	0.3260	1.6010
pmed10	1255	1330.6000	1465.0333	6.0239	16.735	0.7560	2.1003
pmed11	7696	7696.0000	7696.0000	0	0	0	0
pmed12	6634	6634.5666	6648.1333	0.0085	0.2130	0.0056	0.1413
pmed13	4374	4382.1666	4488.8666	0.1867	2.6261	0.0816	1.1486
pmed14	2968	3015.6333	3199.3000	1.6048	7.7931	0.4763	2.3130
pmed15	1729	1837.0333	2010.6666	6.2483	16.2907	1.0803	2.8176
pmed16	8162	8162.0000	8162.3333	0	0.0040	0	0.0033
pmed17	6999	6999.0000	7005.8333	0	0.09760	0	0.0683
pmed18	4809	4822.8666	4944.4333	0.2883	2.8162	0.1386	1.3543
pmed19	2845	2893.6000	3070.9666	1.7082	7.9425	0.486	2.2596
pmed20	1789	1924.5333	2155.7666	7.5759	20.5012	1.3553	3.6676
pmed21	9138	9138.0000	9138.0000	0	0	0	0
pmed22	8579	8579.1333	8591.2000	0.0015	0.1422	0.0013	0.1220
pmed23	4619	4640.4666	4787.8000	0.4647	3.6544	0.2146	1.6880
pmed24	2961	3020.5333	3237.1000	2.0105	9.3245	0.5953	2.7610
pmed25	1828	1971.9666	2185.9000	7.8756	19.5787	1.4396	3.5790
pmed26	9917	9917.0000	9917.4666	0	0.0047	0	0.0046
pmed27	8307	8308.0000	8319.8666	0.012	0.1548	0.0100	0.1286
pmed28	4498	4516.8666	4685.2666	0.4194	4.1633	0.1886	1.8726
pmed29	3033	3088.4666	3331.4666	1.8287	9.8406	0.5546	2.9846
pmed30	1989	2152.9333	2392.2666	8.2419	20.2748	1.6393	4.0326
			Promedio	**1.6135**	**5.3665**	**0.3266**	**1.25254**

De esta forma, los mejores resultados se presentan en la Tabla 20 con los parámetros de la Tabla 21.

Tabla 21. Parámetros finales Ciclo II

Parámetro		Valor
SBHS-LS	Numvecinos ProbabilidadAplicarLS	15 10%
HSOS-LS	Numvecinos ProbabilidadAplicarLS	15 10%

Los resultados obtenidos con los valores de estos parámetros finales (ver Tabla 20), son debido a que el algoritmo adaptado de ascenso a la colina a este problema no es el

adecuado, puesto que el número de movimientos que se realizan sobre la solución son muy pocos para alcanzar un óptimo local.

En una actualización del estado del arte, se encontró una adaptación del operador de cruce mencionado en [26], en el cual mencionaban que este operador de búsqueda local es adecuado para este problema de P-mediana. Por lo tanto, se decidió hacer una adaptación de este operador de cruce para incluirlo en la búsqueda local de los algoritmos adaptados y revisar si se podían lograr mejores resultados.

3.2.5 Adaptación operador cruce

La adaptación de la búsqueda local realizada a partir del operador de cruce encontrado en [26] consta de dos partes: el operador de cruce, que es frecuentemente usado en los algoritmos genéticos, el cual toma 2 individuos de la población de forma aleatoria, en este caso se modificó que estos individuos sean el mejor de la población y la nueva solución generada en cada iteración; y un operador de reparación con conocimiento, en este caso se realizó una reparación aleatoria ya que el conocimiento que se usaba en este operador no se adaptaba al problema.

El algoritmo inicial (ver Figura 13) parte de la solución generada (solución actual) por los algoritmos SBHS y HSOS implementados en el ciclo I y de la mejor solución encontrada en la población actual. Luego, se crea una copia de la solución actual llamada *nueva solución* (línea 3). Después, se itera por cada uno de los puntos de demanda (vértices) de la solución actual y la mejor (línea 4), aquellos vértices donde su estado coincida no se verán afectados (dejando el valor de la solución actual), en cambio en aquellos que sean diferentes y que el mejor de la población esté en estado en 1 (línea 5), se modifica la nueva solución activando su estado (línea 6) y posteriormente se agrega a una lista de genes secundarios las posiciones en donde se modificó la nueva solución (línea 7). En la (línea 8) se repara la solución resultante del cruce y se evalúa en la (línea 9), si el valor de la función objetivo de esa nueva solución es mejor que la solución actual se retorna la nueva solución (líneas 10, 11), en caso contrario se devuelve la solución actual (línea 12).

```
1   INICIO búsqueda_localGN(solucionActual, mejor)
2      lista lista_genes_secundarios
3      nueva_solucion = copiar_solucion(solucionActual)
4      PARA i = 0 HASTA Numvertices
5        SI nueva_solución[i] != mejor[i] y mejor[i] == 1
6           nueva_solucion.activar(i)
7           lista_genes_secundarios.adicionar(i)
8      nueva_solution reparador_busqueda_localGN(nueva_solucion,
9   lista_genes_secundarios)
10     nueva_solución. Evaluar()
11     SI nueva_solucion.Optimo < solución.Optimo
12       RETORNAR nueva_solucion
13     RETORNAR solucionActual
    FIN
```

Figura 13. Búsqueda local GN.

El reparador de cruce (ver Figura 14) recibe la solución actual y la lista de genes secundarios, mientras que la solución actual contenga un número de instalaciones mayor a P (línea 2), se seleccionará aleatoriamente un gen secundario (línea 3) y se inactiva de la

solución (línea 4) para posteriormente eliminarlo de la lista de genes secundarios (línea 5). Cuando la solución contenga el número de instalaciones igual a P, se retorna la solución (línea 6).

```
1  INICIO reparador_busqueda_localGN(solucionActual ,lista_genes_secundarios )
2     MIENTRAS contar(solucionActual.posinstalacions) > p_medianas HACER
3       pos_gen_secundario = aleatorio(lista_genes_secundarios)
4       solucionActual.inactivar(pos_gen_secundario)
5       lista_genes_secundarios.remover(pos_gen_secundario)
6     RETORNAR solucionActual
7  FIN
```

Figura 14. Reparador búsqueda local GN.

El algoritmo al cruzar la nueva solución con la mejor encontrada de la población permitió generar mayor diversificación al ubicar las instalaciones (vértices) de la nueva solución y así encontrar mejores resultados, adicionando el reparador de búsqueda local implementado para obtener un número P de instalaciones (o solución viable), estos resultados se observan en la Tabla 22.

Tabla 22. Resultados con búsqueda local GN

Conjunto de datos		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed1	5819	5819.0000	5819.0000	0.0000	0.0000	0.0000	0.000
pmed2	4093	4093.0000	4095.4000	0.0000	0.0586	0.0000	0.0240
pmed3	4250	4250.0000	4250.0000	0.0000	0.0000	0.0000	0.0000
pmed4	3034	3034.2333	3039.1000	0.0076	0.1680	0.0023	0.0510
pmed5	1355	1361.7333	1396.2000	0.4969	3.0405	0.0673	0.4120
pmed6	7824	7824.0000	7824.0000	0.0000	0.0000	0.0000	0.0000
pmed7	5631	5631.9333	5632.0000	0.0165	0.0177	0.0093	0.0100
pmed8	4445	4445.0000	4456.3666	0.0000	0.2557	0.0000	0.1136
pmed9	2734	2742.7666	2812.2666	0.3206	2.8627	0.0876	0.7826
pmed10	1255	1290.8333	1412.9000	2.8552	12.5816	0.3583	1.5790
pmed11	7696	7696.0000	7696.0000	0.0000	0.0000	0.0000	0.0000
pmed12	6634	6634.0000	6634.0000	0.0000	0.0000	0.0000	0.0000
pmed13	4374	4374.0000	4428.4000	0.0000	1.2437	0.0000	0.5440
pmed14	2968	2971.5666	3132.2000	0.1201	5.5323	0.0356	1.6420
pmed15	1729	1790.0666	1960.4000	3.5319	13.3834	0.6106	2.3140
pmed16	8162	8162.0000	8163.4000	0.0000	0.0171	0.0000	0.0140
pmed17	6999	6999.0000	6999.6000	0.0000	0.0085	0.0000	0.0060
pmed18	4809	4810.9333	4899.1666	0.0402	1.8749	0.0193	0.9016
pmed19	2845	2855.9666	3030.4333	0.3854	6.5178	0.1096	1.8543
pmed20	1789	1862.5666	2103.5000	4.1121	17.5796	0.7356	3.1450
pmed21	9138	9138.0000	9138.0000	0.0000	0.0000	0.0000	0.0000
pmed22	8579	8579.1333	8579.0000	0.0015	0.0000	0.0013	0.0000
pmed23	4619	4619.6666	4754.3000	0.0144	2.9292	0.0066	1.3530
pmed24	2961	2972.2333	3192.9000	0.3793	7.8318	0.1123	2.3190
pmed25	1828	1911.6333	2157.6000	4.5751	18.0306	0.8363	3.2960

Conjunto de datos		**Promedio**		**FmedianRPE**		**Desviación estándar**	
Problema	Óptimo	HSOS	SBHS	HSOS	SBHS	HSOS	SBHS
pmed26	9917	9917.0000	9917.7000	0.0000	0.0070	0.0000	0.0070
pmed27	8307	8307.3333	8307.2000	0.0040	0.0024	0.0033	0.0020
pmed28	4498	4502.7333	4637.5333	0.1052	3.1021	0.0473	1.3953
pmed29	3033	3039.9000	3294.2333	0.2274	8.6130	0.0690	2.6123
pmed30	1989	2085.5000	2357.0333	4.8516	18.5034	0.9650	3.6803
			Promedio	**0.7348**	**4.1387**	**0.1359**	**0.9352**

Teniendo un error relativo porcentual menor al encontrado en el Ciclo I (ver Tabla 23), se obtienen mejores resultados cruzando los puntos de demanda de la nueva solución con el mejor de la población, mostrando que este operador de cruce es adecuado para el problema de FLP.

Tabla 23 Comparación Ciclo I y Ciclo II

		Ciclo I		**Ciclo II**	
Problema		FmedianRPE		FmedianRPE	
Conjunto de datos	Óptimo	SBHS	HSOS	SBHS	HSOS
pmed1	5819	0	0	0.0000	0.0000
pmed2	4093	0.0488	0	0.0586	0.0000
pmed3	4250	0	0	0.0000	0.0000
pmed4	3034	0.6141	0.0219	0.1680	0.0076
pmed5	1355	4.3099	0.9717	3.0405	0.4969
pmed6	7824	0	0	0.0000	0.0000
pmed7	5631	0.0047	0.0248	0.0177	0.0165
pmed8	4445	0.5331	0	0.2557	0.0000
pmed9	2734	3.4515	0.5327	2.8627	0.3206
pmed10	1255	13.7078	4.0371	12.5816	2.8552
pmed11	7696	0	0	0.0000	0.0000
pmed12	6634	0	0	0.0000	0.0000
pmed13	4374	1.5889	0.0076	1.2437	0.0000
pmed14	2968	5.8883	0.4368	5.5323	0.1201
pmed15	1729	13.5935	4.3551	13.3834	3.5319
pmed16	8162	0	0	0.0171	0.0000
pmed17	6999	0.0180	0	0.0085	0.0000
pmed18	4809	2.1286	0.0478	1.8749	0.0402
pmed19	2845	6.8916	0.69	6.5178	0.3854
pmed20	1789	17.4380	4.8481	17.5796	4.1121
pmed21	9138	0	0	0.0000	0.0000
pmed22	8579	0	0	0.0000	0.0015
pmed23	4619	3.0367	0.0577	2.9292	0.0144
pmed24	2961	8.0321	0.7992	7.8318	0.3793
pmed25	1828	18.5138	5.423	18.0306	4.5751
pmed26	9917	0.0047	0	0.0070	0.0000
pmed27	8307	0.0012	0.0012	0.0024	0.0040
pmed28	4498	3.3800	0.1494	3.1021	0.1052

		Ciclo I		**Ciclo II**	
Problema		FmedianRPE		FmedianRPE	
pmed29	3033	8.8119	0.5879	8.6130	0.2274
pmed30	1989	18.8536	5.6259	18.5034	4.8516
	Promedio	4.3616	0.9539	**4.1387**	**0.7348**

Capítulo 4

4 ALGORITMOS ADAPTADOS: HSOS y SBHS

En este capítulo se presenta la representación de las soluciones candidatas, la función objetivo utilizada en el problema de ubicación de facilidades de *P*-Mediana, las adaptaciones y las mejoras realizadas a los algoritmos SBHS y HSOS. Por último, el esquema general del proceso e implementación desarrollada para ubicar instalaciones.

4.1 Representación de las soluciones

Cuando se trabaja con algoritmos metaheurísticos es necesario representar el problema en una solución del espacio de búsqueda, de manera que se pueda mapear cada objeto del espacio como una solución candidata (codificación), y a partir de la solución candidata, su correspondiente ubicación en el espacio de búsqueda. De esta manera se usa la codificación en la construcción de las soluciones y a partir de esto hacer una evaluación de las soluciones candidatas.

Teniendo en cuenta lo expuesto anteriormente en los algoritmos SBHS y HSOS adaptados, la representación de dicha solución para el problema de ubicación de instalaciones, se realiza mediante una codificación binaria en forma de vector; de tamaño n, que equivale a la cantidad de posibles ubicaciones en el espacio geográfico (vértices), es decir: $S = [V_1, V_2, \dots V_k, \dots, V_n]$, donde cada elemento V_k del vector toma un valor de 0 o 1, donde 0, representa que no se ha ubicado una instalación en ese punto de demanda (vértice) del espacio geográfico, y 1 que está ubicada una instalación en este punto. Por ejemplo, si la cantidad de posibles ubicaciones es $diez$ $(n = 10)$, y el número de instalaciones a ubicar es 2 $(p = 2)$, la representación de una solución podría ser la siguiente $S = [1,0,0,0,1,0,0,0,0,0]$, lo que significa que en la posición 1 y 5 (vértice) del vector se ubicará una instalación.

4.2 Función objetivo

El valor de la función objetivo (ver ecuación (20)), es un valor entero igual a la sumatoria de la mínima distancia entre un punto de demanda i hacia las instalaciones m_j. Teniendo en cuenta que la función tiene una restricción sobre el número de instalaciones P que serán ubicadas en el espacio geográfico [5].

$$f_i^k = f\left(S_i^k\right) = \sum_{i=1}^{NP} \left(\min\left(dist[i]\left[m_j\right]\right)\right) \tag{20}$$

$$m_j \in \{m_1, m_2, m_3, \dots, m_p$$

En este caso, se requiere minimizar el valor de esta función objetivo para los puntos de demanda que están en el espacio geográfico, y para la cual está compuesta por las siguientes características: vértice, es la posición del punto de demanda en el espacio geográfico y puede ser seleccionada para ubicar una instalación; arista, es la distancia entre los vértices, es decir, entre los puntos de demanda o hacia una instalación; y P, que es el número de instalaciones a ubicar en el espacio.

4.3 Algoritmo HSOS

Las modificaciones realizadas al algoritmo de organismos simbióticos para la adaptación al problema de ubicación de instalaciones *P*_mediana no capacitada, fueron: el método de inicialización y reparación de la población; se agrega en las fases de dicho algoritmo (excepto en la fase de improvisación) límites para la activación o cierre de instalaciones de una solución, debido a que en algunas ocasiones la solución resultante, al aplicar las fases, no ubica alguna instalación en el espacio geográfico (límite inferior), y a que la reparación utilizada sobre estos algoritmos necesita al menos que una instalación este ubicada, así mismo, un límite que no permita a la solución ubicar muchas instalaciones debido a la complejidad computacional (límite superior.); por último, en cada una de las fases del algoritmo pasa por un método de reparación distinto al planteado en los algoritmos base y se añade un algoritmo de búsqueda local que se explicara al final de esta sección. Estas modificaciones se encuentran en negrita en el pseudocódigo del algoritmo HSOS (ver Figura 15).

El algoritmo HSOS inicializa la población de organismos haciendo uso del operador de inicialización con conocimiento (línea 6) aplicando el método de reparación con conocimiento (ver sección 3.1.5.1), e identifica al mejor individuo de la población (línea 8).

Este algoritmo itera hasta que el número de $Efos$ sea menor o igual a $MaxEfos$ o hasta que el óptimo sea encontrado. Luego en cada iteración, cada individuo de la población es sometida a distintas fases que realizan procesos de evolución de la siguiente forma: en la fase de mutualismo busca modificar la solución actual (X_i) y una solución que es seleccionada de la población de forma aleatoria (X_j), beneficiando en la primera parte (línea 12) a la solución escogida de forma aleatoria, por medio de la solución actual, y en la segunda parte (línea 18) a la solución actual por medio de la solución escogida de forma aleatoria, en la fase de comensalismo sustituye la solución actual beneficiándose de una solución escogida aleatoriamente (línea 26) ,en la fase de parasitismo genera un parasito a partir de la solución actual y reemplaza la escogida aleatoriamente (línea 33), y en la fase de improvisación (línea 40) se crea una nueva solución a partir de la población y sustituye el peor de la población con la nueva solución.

Cada una de las fases es modificada por una búsqueda local (ver Figura 15), y después, en la fase de mutualismo se reemplaza la solución actual en la población o la seleccionada aleatoriamente (líneas 14-16 y 20-22); en comensalismo, se sustituye la solución actual (líneas 28-30); en parasitismo se reemplaza la solución seleccionada aleatoriamente de la población (líneas 36-38), y en improvisación se reemplaza al peor de la población (líneas 43-45).

```
1   INICIO
2         MiProblema = Problema
3         MiAleatorio = Aleatorio
4         Efos = 0
5         MaxEfos = 1500
6         Poblacion = InicializarPoblacionConocimiento(HMS)
7         Mejor = Utilidades.ObtenerMejor(Poblacion)
8         MIENTRAS Efos < MaxEfos && Mejor.Optimo > MiProblema.Optimo
9                 PARA i = 0 HASTA Tamaño(Poblacion)
10                      j = aleatorio(!=i)
11                      //Mutualismo
12                      m1 = mutualismo(Poblacion[j], Poblacion[i], mejor)
13                      m1 = Busquedalocal(m1)
14                      SI m1.Optimo < Poblacion[j].Optimo ENTONCES
15                              Poblacion[j] = m1
16                      FIN SI
17                      //Mutualismo
18                      m2 = mutualismo(Poblacion[i],Poblacion[j], mejor)
19                      m2 = Busquedalocal(m2)
20                      SI m2.Optimo < Poblacion[i].Optimo ENTONCES
21                              Poblacion[i] = m2
22                      FIN SI
23
24                      //Comensalismo
25                      j = aleatorio(!=i)
26                      m3 = comensalismo(Poblacion[i], Poblacion[j], mejor)
27                      m3 = Busquedalocal(m3)
28                      SI m3.Optimo < Poblacion[i].Optimo ENTONCES
29                              Poblacion[i] = m3
30                      FIN SI
31                      //Parasitismo
32                      j = aleatorio(!=i)
33                      m4 = Parasitismo(Poblacion[i])
34                      m4 = Busquedalocal(m4)
35
36                      SI m4.Optimo < Poblacion[j].Optimo ENTONCES
37                              Poblacion[j] = m4
38                      FIN SI
39                      //Improvisacion
40                      m5 = Improvisacion(i)
41                      m5 = Busquedalocal(m5)
42
43                      SI m5.Optimo < Peor(Poblacion) ENTONCES
44                              Poblacion[Peor] = m5
45                      FINSI
46                      mejor=ObtenerMejor(Poblacion)
47                FIN PARA
48        FIN MIENTRAS
49  FIN
```

Figura 15 Pseudocódigo HSOS.

La fase de mutualismo (ver Figura 16), recibe como parámetros la solución a modificar (X_i), la solución que es usada para la modificación (X_j) y la mejor de la población ($mejor$). En esta fase se inicializa de forma aleatoria una solución $VectorMutuo$ de acuerdo con una probabilidad equitativa del 50%, que determina si el valor de cada elemento en el $VectorMutuo$ es obtenida de los vectores X_i o X_j (líneas 2-9); luego se inicializa una solución A de forma aleatoria (líneas 10 y 11), a esta solución se le realiza una reasignación del valor en sus vértices en donde la solución $VectorMutuo$ y la mejor solución de la

población coincidan en su dimensión $(1 \text{ o } 0)$ (Líneas 12-15). De igual forma se genera una solución B de forma aleatoria (Línea 17-18), la cual sufre una reasignación en los valores de los vértices si A coincide con el organismo X_j (ver líneas 22-23). Teniendo en cuenta que la solución B puede obtener una asignación donde el número de instalaciones sea menor a $limInf$, esta solución activa con respecto al valor que tiene la solución A (línea 24), en caso de que tenga demasiadas instalaciones $limSup$, desactiva con respecto a A (línea 26); estas líneas de código actúan de igual forma para las siguientes fases. En caso de que el número de instalaciones se encuentre entre los límites, entonces la solución asigna los valores de sus vértices con respecto a A (líneas 27-29). Por último, repara la solución hasta que las instalaciones asignadas en la solución B sean igual a P (Línea 33), evalúa la función objetivo para esta solución (Línea 34) y luego la retorna (Línea 35).

```
1   INICIO Mutualismo(solución Xi, solucion Xj, solucion mejor)
2     Vectormutuo
3     DESDE k = 0 HASTA NumVertices HACER
4       SI aleatorio(0,1) > 0.5 ENTONCES
5         Vectormutuo [k] = Xj.Vertices[k]
6       SINO
7         Vectormutuo [k] = Xi.vertices[k]
8       FIN SI
9     FIN DESDE
10    A = nueva Solucion(MiAlgoritmo)
11    A.InicializacionAleatoria()
12    DESDE k = 0 hasta NumVertices HACER
13      SI Vectormutuo [k] == mejor.Vertices[k] ENTONCES
14        A.Vertices[k] = mejor.Vertices[k]
15      FIN SI
16    FIN DESDE
17    B = nueva Solucion(MiAlgoritmo)
18    B.InicializacionAleatoria()
19    limInf = MiAlgoritmo.MiProblema.Pmedianas * 0.50
20    limSup = MiAlgoritmo.MiProblema.Pmedianas * 1.50
21    DESDE k = 0 hasta NumVertices HACER
22      SI A.Vertices[k] == Xi.vertices[k] ENTONCES
23        SI ContarInstalaciones(B) < limImf ENTONCES
24          SI A.Vertices[k] == 1 ENTONCES B.Activar(k)
25        SINO SI ContarInstalaciones(B) > limSup ENTONCES
26          SI A.Vertices[k] == 0 ENTONCES B.Inactivar(k)
27        SINO
28          SI A.Vertices[k] == 1 ENTONCES B.activar(k)
29          SINO B.Inactivar(k)
30        FIN SI
31      FIN SI
32    FIN DESDE
33   B.RepararConconocimiento()
34   Evaluar(B)
35   Retornar B
36  FIN
```

Figura 16. Pseudocódigo Mutualismo

En la fase de comensalismo (ver Figura 17) se recibe como parámetro la solución actual (X_i), una solución que es seleccionada de la población de forma aleatoria (X_j) y la mejor de la población $(Mejor)$. Esta fase inicializa una solución c aleatoriamente usando como base una asignación aleatoria (ver línea 3), esta solución reasigna los valores en sus vértices si el organismo X_j y la $Mejor$ solución coincide en su dimensión (línea 4-7). De igual forma,

se inicializa una solución d (líneas 9 y 10), reasignando aquellos vértices sobre la solución si el organismo X_i y la solución c coincidan (línea 13-22). En esta reasignación se tiene en cuenta que en la solución d este entre los límites inferior y superior de instalaciones para obtener un número de instalaciones adecuado (ver líneas 11 y 12), haciendo la validación como se explicó en la fase de mutualismo. Por último, se repara, evalúa y retorna la solución d (líneas 23-25).

```
1   INICIO Comensalismo(solución xi, solución xj, solución mejor)
2     c = nueva Solucion(MiAlgoritmo)
3     c.Inicializacionaleatoria()
4     DESDE k = 0 HASTA NumVertices
5       SI xj.Vertices[k] == mejor.Vertices[k] ENTONCES
6         c.Vertices[k] = mejor.Vertices[k]
7       FIN SI
8     FIN DESDE
9     d = nueva Solucion(Mialgoritmo)
10    d.Inicializacionaleatoria()
11    limInf = MiAlgoritmo.MiProblema.Pmedianas * 0.50
12    limSup = MiAlgoritmo.MiProblema.Pmedianas * 1.50
13    DESDE k = 0 HASTA NumVertices
14      SI c.Vertices[k] == xi.vertices[k] ENTONCES
15        SI ContarInstalaciones(d) < limImf ENTONCES
16          SI c.Vertices[k] == 1 ENTONCES d.Activar(k)
17        SINO SI ContarInstalaciones(c) > limSup ENTONCES
18          SI c.Vertices[k] == 0 ENTONCES d.Inactivar(k)
19        SINO
20          SI c.Vertices[k] == 1 ENTONCES d.Activar(k)
21          SINO d.Inactivar(k)
22    FIN DESDE
23   d.RepararConConocimiento()
24   d.Evaluar()
25   retornar d
26  FIN
```

Figura 17. Pseudocódigo Comensalismo.

En la fase de parasitismo (ver Figura 18) recibe como parámetro la solución actual (X_i) y crea una solución parasito a partir de los vértices de esta (línea 2), posteriormente el parasito sufre una reasignación del estado en sus vértices de acuerdo con una lista de instalaciones seleccionadas de forma aleatoria (línea 3), que permitirá el cambio de estado sobre este vector (línea 6). Al igual que en las anteriores fases en esta también se incluye los límites inferior y superior, para que el $parasito$ obtenga un número adecuado de instalaciones (líneas 7-14), si el número de instalaciones se encuentra entre los límites, entonces la solución reasigna los valores de sus vértices con respecto al mismo $parasito$ (líneas 11-13); posteriormente repara, evalúa y retorna la solución $parasito$ (líneas 16-18).

```
1   INICIO Parasitismo(solucion xi)
2    Parasito = copiasolucion(xi)
3    numVertices = SeleccionarVerticesAleatoriamente()
4    limImf = MiAlgoritmo.MyProblema.Pmedianas * 0.50
5    limSup = MiAlgoritmo.MyProblema.Pmedianas * 1.50
6    PARA Cada posición en numVertices HACER
7        SI(contarInstalaciones(parasito) < limImf) ENTONCES
8          SI parasito.Vertices[posicion] == 0 ENTONCES parasito.Activar(posicion)
9        SINO SI ContarInstalaciones(parasito) > limSup ENTONCES
10         SI parasito.Vertices[posición] == 1 ENTONCES parasito.activar(Posicion)
11       SINO
12         SI parasito.Vertices[posicion] == 1 ENTONCES parasito.Inactivar(posicion)
13         SINO parasito.activar(Posicion)
14     FIN SI
15   FIN PARA
16   Parasito.RepararConconocimiento()
17   Evaluar(Parasito)
18   Retornar Parasito
19  FIN
```

Figura 18. Pseudocódigo Parasitismo.

En la fase de Armonía (ver Figura 19) recibe como parámetro la posición de la solución actual en la población (X_i) y hace uso de diferentes probabilidades de acuerdo con los parámetros ($PAR\ y\ HMCR$) para determinar los estados de la nueva solución ($Neko$). La primera condición (línea 6) valida si un valor aleatorio entre 0 y 1 es menor al valor de $HMCR$, si se cumple esta condición, para la dimensión (vértice) de la nueva solución (vector $Neko$) se selecciona aleatoriamente una solución de la población que sea diferente a la solución actual xj (línea 7-10) y se asigna el valor que esta solución tiene en la misma dimensión (línea 11). Además, de acuerdo con la segunda condición se determina la probabilidad de que un valor aleatorio sea menor o igual al valor PAR (línea 12), si se cumple esta condición, se realiza una reasignación en la dimensión actual del vector $Neko$ (línea 13-17), de acuerdo con una probabilidad del 50%. En caso de que la primera condición no se cumpla, se realiza una asignación de estado a la dimensión actual del vector $Neko$ de forma aleatoria (línea 20). Por último, se repara, evalúa y retorna la solución $Neko$ (líneas 23-25).

```
1   INICIO Improvisacion(pos Xi)
2     Par = 0.5
3     Hmcr = 0.95
4     Neko = nueva Solucion()
5     DESDE K = 0 HASTA NumVertices
6       SI aleatorio(0,1) < hmcr ENTONCES
7         posAleatoria = MiRandom (0, tamañoPoblacion)
8         MIENTRAS posAleatoria == Xi HACER
9            posAleatoria = MiRandom (0, tamañoPoblacion)
10        FIN MIENTRAS
11        Neko.Vertices[k] = poblacion[posAleatoria].Vertices[k]
12        SI aleatorio(0,1) <= par ENTONCES
13          SI aleatorio(0,1) > 0.5 ENTONCES
14            Neko.Vertices[k] = 0
15          SINO
16            Neko.Vertices[k] = 1
17          FIN SI
18        FIN SI
19      SI NO
```

```
20          Neko.Vertices[k] = EstadoAleatorio()
21        FIN SI
22      FIN DESDE
23      ReparарConconocimiento(Neko)
24      Evaluar(Neko)
25      Retornar Neko
26    FIN
```

Figura 19. Pseudocódigo Improvisación.

Por último, se implementó la búsqueda local (ver Figura 20) que recibe como parámetro la solución actual (X_i) y la mejor solución encontrada en la población actual ($Mejor$). Luego, se crea una copia de la solución actual llamada *nueva solución* (línea 3). Después, se itera por cada uno de los puntos de demanda (vértices) de la solución actual y la mejor (línea 4), aquellos vértices donde su estado coincida no se verán afectados (dejando el valor de la solución actual), en cambio en aquellos que sean diferentes y que el mejor de la población esté en estado en 1 (línea 5), se modifica la nueva solución activando su estado (línea 6) y se agrega a una lista de genes secundarios las posiciones en donde se modificó la nueva solución (línea 7). En la (línea 8) se repara la solución resultante del cruce y se evalúa en la (línea 9), si el valor de la función objetivo de esa nueva solución es mejor que la solución actual se retorna la nueva solución (líneas 10-11), en caso contrario se devuelve la solución actual (línea 12).

```
1   INICIO búsqueda_localGN(solucion, mejor)
2      lista lista_genes_diferentes
3      nueva_solucion = copiar_solucion(solucion)
4      PARA i = 0 HASTA Numvertices
5        SI nueva_solución[i] != mejor[i] AND mejor[1] == 1
6           nueva_solucion.activar(i)
7           lista_genes_secundarios.adicionar(i)
8      nueva_solution reparador_busqueda_localGN(nueva_solucion, lista_genes_secundarios)
9      nueva_solución. Evaluar()
10     SI nueva_solucion.Fitness < solución.Fitness
11       RETORNAR nueva_solucion
12     RETORNAR nueva_solucion
13   FIN
```

Figura 20 Implementación Búsqueda Local HSOS.

En la Tabla 24, se presenta un resumen de las modificaciones realizadas al algoritmo de HSOS y la razón de cada una de ellas.

Tabla 24 Modificaciones HSOS

Modificaciones HSOS		
Modificación/ Adición	Lugar del algoritmo	Motivo
Adición de parámetros: LimInf, LimSup	Fases de mutualismo, parasitismo y comensalismo.	No obtener soluciones inviables y reducir tiempos de ejecución del algoritmo.
Adición de Búsqueda local	Después de las fases de mutualismo, parasitismo, comensalismo e improvisación.	Tratar de mejorar cada organismo resultante de las fases en cada iteración

Modificación de la Inicialización	Inicialización de la población.	Decrementar el número de instalaciones activas en cada solución al inicializar la población, para disminuir el tiempo en la reparación de las soluciones.
Modificación del operador de Reparación	Después de las fases de mutualismo, parasitismo, comensalismo e improvisación.	Obtener el numero *P* de instalaciones en la solución de acuerdo con el problema.

4.4 Algoritmo SBHS

Las modificaciones realizadas al algoritmo de búsqueda armónica simplificada fueron: el método de inicialización; un parámetro límite en la adición de instalaciones, debido a que en algunas ocasiones la solución resultante al crear una nueva solución no ubica alguna instalación en el espacio geográfico; el método de reparación, el cual es distinto al planteado en el algoritmo base; y un algoritmo de búsqueda local.

El algoritmo SBHS adaptado al problema de ubicación de instalaciones *P*_mediana no capacitado (ver Figura 21), inicializa los parámetros necesarios según el problema (líneas 2 a 7), inicializa la población con el método de conocimiento (ver sección 3.1.4.2) (línea 8) siendo la población ordenada (línea 9) y selecciona la mejor solución de la población (línea 10). Luego, el algoritmo itera creando una nueva armonía teniendo en cuenta que si un aleatorio es menor a la tasa de consideración de la memoria (HMCR) (línea 14) almacenando el valor de 0 o 1 en la variable $estado$ de acuerdo con la fórmula que está en la línea 17, si el valor de la variable estado es igual a 1 (línea 19), entonces activa una instalación (línea 20), en caso contrario si el número de instalaciones prendidas es menor al $liminf$ también la activa (línea 22-23) y de esta forma no obtener soluciones inviables; si el aleatorio es mayor a HMCR, activa una instalación a partir de una probabilidad del 50% siguiendo la restricción del $liminf$ (líneas 27-32).

Por último, esta solución es reparada con conocimiento (ver sección 3.1.5.1) y sometida al operador de búsqueda local (línea 36-37), y si el valor de la función objetivo de la nueva armonía es mejor al peor de la población, esta es reemplazada (línea 40).

```
1   INICIO SBHS()
2     MiProblema = Problema;
3     MiAleatorio = Aleatorio;
4     //Ajuste de parámetros
5     hmcr = 0.95
6     Efos = 0
7     MaxEfos = 1500
8     poblacion = InicializarPoblacionConConocimiento(tamañoPoblacion)
9     Ordenar(poblacion)
10    mejor = Poblacion[0]
11    Mientras Efos < MaxEfos y mejor.Fitness > Problema.Optimo HACER
12      nuevaArmonia = nuevaSolucion();
13      DESDE i = 0 HASTA NumVertices HACER
14        SI(MiAleatorio (0,1) <= HMCR) HACER
15          Pos1 = MiAleatorio (0, tamañoPoblacion)
```

```
        Pos2 = MiAleatorio (0, tamañoPoblacion) si Pos2 != Pos1
        estado = poblacion[pos1].vertice[i] + -1^(Poblacion[pos1].vertice[i]) *
|mejor.vertice[i]- Poblacion[pos2].vertice[i]|
        SI estado = 1 ENTONCES
          nuevaArmonia.activar(i)
        SINO
          SI hayPrendidas(nuevaArmonia) < liminf ENTONCES
            nuevaArmonia.activar(i)
          FIN SI
        FIN SI
      SINO
        SI MiAleatorio (0,1) < 0.5 ENTONCES
          SI hayprendidas(nuevaArmonia) < liminf ENTONCES
            nuevaArmonia.activar(i)
          FIN SI
        SINO
          nuevaArmonia.activar(i)
        FIN SI
      FIN SI
    FIN DESDE
    ReparарConconocimiento(nuevaArmonia)
    BusquedaLocal(nuevaArmonia)
    Evaluar(nuevaArmonia)
    SI nuevaArmonia.Optimo < peor(poblacion).Optimo
      Peor(poblacion) = nuevaArmonia
    FIN SI
  FIN MIENTRAS
FIN
```

Figura 21. Pseudocódigo SBHS.

La búsqueda local implementada es la misma del algoritmo HSOS que se muestra en la Figura 20, y explicado anteriormente. Un resumen de las modificaciones realizadas al algoritmo de SBHS y la razón de cada una de ellas se muestran en la Tabla 25.

Tabla 25 Modificaciones SBHS

Modificaciones SBHS		
Modificación/ Adición	Lugar del algoritmo	Motivo
Adición de parametros: LimInf	Generación de una nueva solución	No obtener soluciones inviables.
Adición de Búsqueda local	Generación de una nueva solución	Tratar de mejorar la nueva solución al aplicar la formula simplificada.
Modificación de la Inicialización	Inicialización de la población.	Decrementar el número de instalaciones activas en cada solución al inicializar la población, para disminuir el tiempo en la reparación de las soluciones.
Modificación del operador de Reparación	Inicialización y evolución.	Obtener el numero *P* de instalaciones en la solución de acuerdo con el problema.

4.5 Características generales de la implementación

Para el desarrollo de este proyecto, se optó por una ejecución distribuida, teniendo en cuenta que los algoritmos se debían ejecutar 30 veces para cada una de las instancias de los conjuntos de datos de los problemas FLP, y el tiempo de procesamiento era muy alto, debido a la alta dimensionalidad y número de medianas a ubicar de estos problemas.

La arquitectura distribuida planteada usa servicios web para almacenar los datos dados por la aplicación (ver Figura 22), logrando un acceso más rápido a estos. Esta arquitectura ejecuta los algoritmos en el cliente, el cual se ejecuta en 18 computadores y 10 máquinas virtuales para minimizar los tiempos de ejecución de cada problema, estos se conectan a una aplicación servidor que accede a las tablas creadas en la base de datos para conocer que algoritmo y con qué semilla se debe ejecutar, enviándolo como tarea al cliente. De igual forma, salva los resultados obtenidos cuando se finaliza la tarea por parte del cliente, por último, se tiene las tablas creadas en un servidor en la nube para ser accedida por la aplicación servidor.

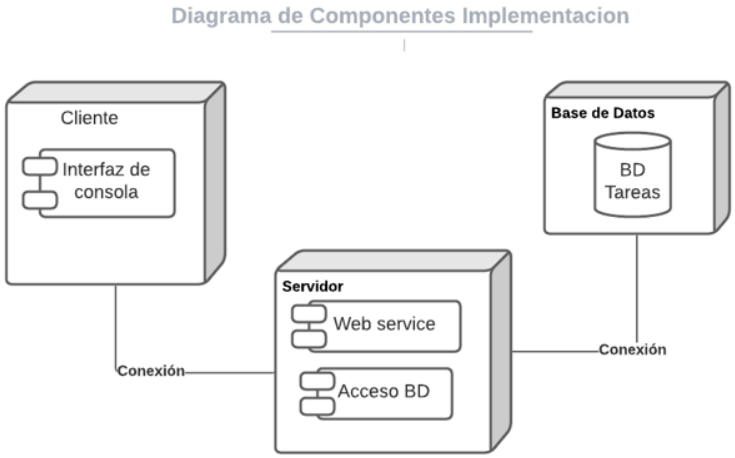

Figura 22. Diagrama componentes Implementación.

Posteriormente, se identificó cada una de las clases necesarias para la implementación de este proyecto (ver Figura 23), teniendo una clase *solution* que contiene todos los métodos en común que usan los algoritmos implementados en la solución, *utils* usado para definir métodos usados en la aplicación, *Pmediana* y *Arista* que permite la adaptación de los algoritmos al problema, y la interacción de los algoritmos implementados con la arquitectura desarrollada, *Algorithm* que ejecuta los algoritmos implementados y además contiene métodos para inicializar la población, por último se tiene las clases de los algoritmos implementados que contienen los parámetros necesarios para ser ejecutados.

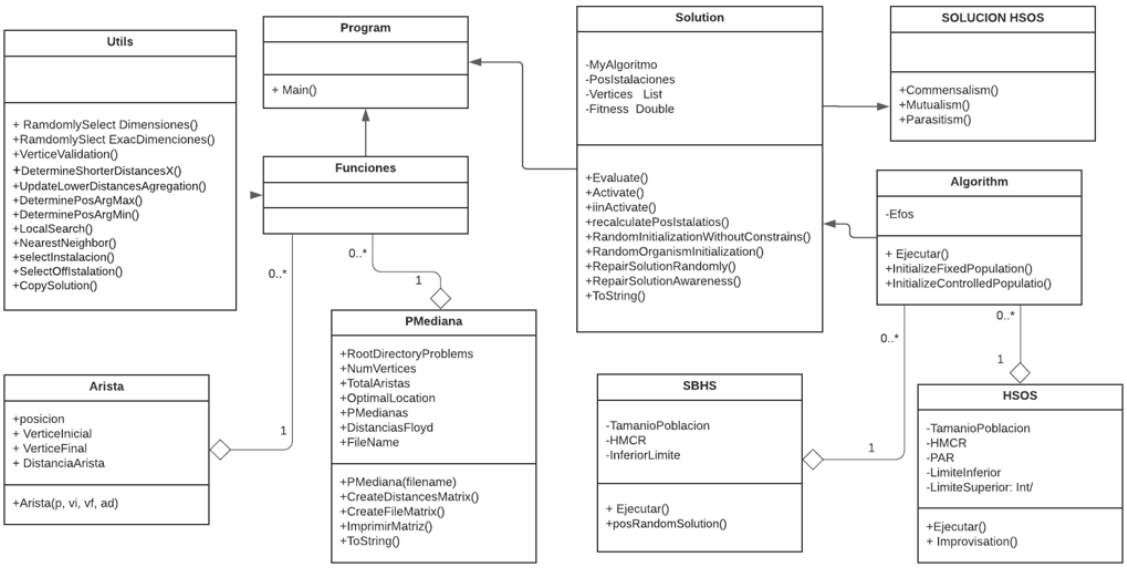

Figura 23. Diagrama de clases cliente.

Además, se creó una aplicación servidor que contiene los métodos para almacenar los datos procesados por el cliente y acceder a que tarea se debe ejecutar, la aplicación lee los datos de las tablas para saber que algoritmo se ha ejecutado y con qué semilla (*GetTask*), de esta forma obtiene la tarea a ejecutar y posteriormente salva (*SaveResults*) los resultados obtenidos (ver Figura 24).

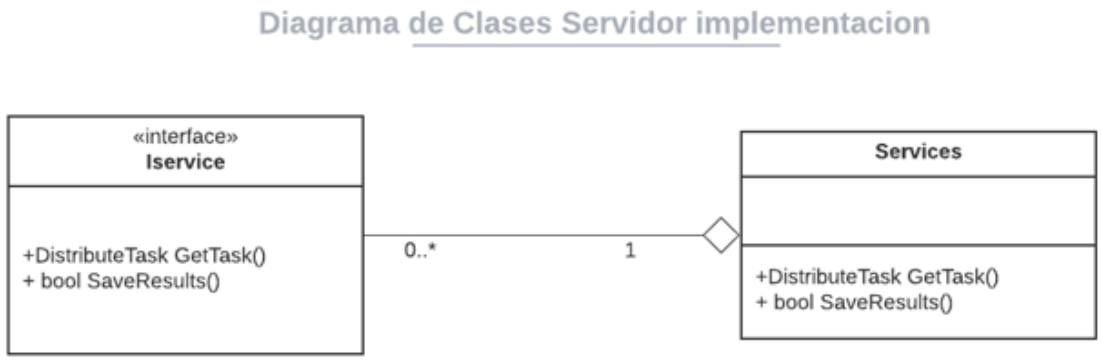

Figura 24. Diagrama de clases Servidor.

Y por último se generaron 2 tablas para almacenar la información necesaria para la ejecución de los problemas y de esta forma guardar la información de las soluciones obtenidas al hacer ejecución de estas 30 veces para cada instancia, así como su estado e información importante para la ejecución por parte de los algoritmos para cada una de ellas. Esta estructura se detalla en la Figura 25.

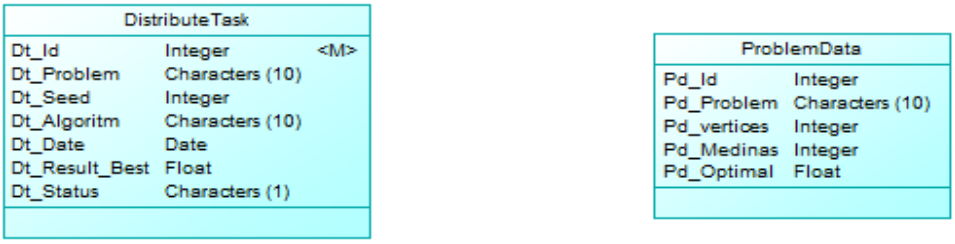

Figura 25. Diagrama de tablas de información.

Capítulo 5

5 EVALUACIÓN

Este capítulo se describe los conjuntos de datos y las métricas usadas en el proceso experimental llevado a cabo para realizar la evaluación de los resultados obtenidos por los dos algoritmos adaptados en este proyecto. También, los valores definidos para los parámetros de cada uno de los algoritmos. Por último, la comparación de los resultados de los algoritmos adaptados, frente a otros algoritmos del estado del arte.

5.1 Conjunto de datos

OR-Library[3] es un repositorio que contiene el subconjunto de datos de prueba usado en este proyecto para problemas de investigación de operaciones (OR), entre los cuales, se encuentra el conjunto de datos *P*_med que busca tener problemas para la ubicación de instalaciones de *P*_mediana no capacitado, que comprenden los archivos con los siguientes nombres: Pmed1 al Pmed30, respectivamente.

Los datos de cada uno de los problemas (conocidos como instancias) se encuentran en un archivo plano con filas de tres columnas con el siguiente formato (Ver Figura 26): la primera fila presenta el número de vértices para el problema (400), número de aristas (3200) y el número P (40) de instalaciones a ubicar; el resto de las filas indica la conexión entre un vértice A y un B, y la distancia que hay entre ellos.

400	**3200**	**40**
1	2	66
2	3	64
3	4	1
4	5	28
5	6	32
6	7	5
7	8	92
8	9	7
9	10	82
10	11	60

Figura 26 Formato conjunto de datos

[3] Instancias tomadas en la siguiente URL: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/

5.2 Descripción de las métricas

Las métricas consideradas en este proyecto miden la calidad de la solución encontrada para cada algoritmo. Una de estas métricas está basada en la función de porcentaje de error relativo (RPE) [5] y su fórmula se presenta en la ecuación (21).

$$f\ median\ RPE = \frac{fmedian - fopt}{fopt} * 100 \tag{21}$$

Donde, $fmedian$ es el promedio del valor obtenido por la función objetivo de las 30 ejecuciones del algoritmo, $fopt$ es el valor óptimo para cada conjunto de datos.

La otra métrica es la desviación estándar cuya forma de cálculo se presenta en la ecuación (22)

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n}(X_i - X)^2}{n}} \tag{22}$$

Donde, X_i, representa la solución en su ejecución i-esima, n el número total de ejecuciones sobre un conjunto de datos y X representa el promedio de la solución al haber ejecutado n veces el algoritmo.

5.3 Afinamiento de parámetros

El proceso de afinamiento se explicó anteriormente en la sección 3.1.7, el cual permitió definir los valores de los parámetros para los algoritmos HSOS y SBHS, como se muestra en la Tabla 26.

Tabla 26 Parámetros finales SBHS y HSOS con LS

Parámetro		Valor
SBHS	HMS HMCR LimInf	25 0.95 NumVertices*0.25
HSOS	HMS PAR HMCR LimInf LimSup	25 0.5 0.95 NumVertices*0.50 NumVertices*1.50

El algoritmo de búsqueda local implementado no tiene parámetros, por este motivo, se incrementó el número de evaluaciones de la función objetivo utilizada en el Capítulo III obteniendo un menor porcentaje de error relativo y en la desviación.

5.4 Evaluación

Para realizar la evaluación de los algoritmos adaptados HSOS y SBHS, se compararon los resultados obtenidos con algoritmos del estado del arte que hacen uso del mismo conjunto de datos. El algoritmo ABC [5] es una metaheurística usada para resolver problemas de optimización que busca a partir del proceso que realizan las abejas en la recolección de las fuentes de néctar crear soluciones al hacer una exploración; y el algoritmo TS [1] que añade a la lista Tabú posibles soluciones a partir de movimientos que se hace sobre la solución actual y elimina aquellas que no han mejorado después de un número de iteraciones.

El algoritmo HSOS obtuvo un resultado igual al óptimo (ver Tabla 27), en los problemas: Pmed1, Pmed2, Pmed3, Pmed6, Pmed8, Pmed11, Pmed12, Pmed13, Pmed16, Pmed17, Pmed21 y Pmed26; un mejor resultado que el algoritmo TS en el conjunto de datos: Pmed4, Pmed9, Pmed18, Pmed19, Pmed28; y un mejor resultado al algoritmo ABC en el conjunto de datos Pmed18. Además, logró acercarse al óptimo de los problemas: Pmed7, Pmed14, Pmed22, Pmed23, Pmed27 y Pmed 29.

Tabla 27. Resultados HSOS

Conjunto de datos				**FMEDIAN RPE**			**Desviación estándar**		
Problema	n	P	Óptimo	HSOS	ABC	TS	HSOS	ABC	TS
pmed1	100	5	5819	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed2	100	10	4093	0.0000	0.0000	0.2931	0.0000	0.0000	0.1200
pmed3	100	10	4250	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed4	100	20	3034	0.0076	0.0000	0.3955	0.0023	0.0000	0.1200
pmed5	100	33	1355	0.4969	0.0000	0.0000	0.0673	0.0000	0.0000
pmed6	200	5	7824	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed7	200	10	5631	0.0165	0.0000	0.1420	0.0093	0.0000	0.0800
pmed8	200	20	4445	0.0000	0.0000	0.2024	0.0000	0.0000	0.0900
pmed9	200	40	2734	0.3206	0.1500	0.6949	0.0876	0.0410	0.1900
pmed10	200	67	1255	2.8552	0.7200	0.4780	0.3583	0,0903	0.0600
pmed11	300	5	7696	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed12	300	10	6634	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed13	300	30	4374	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed14	300	60	2968	0.1201	0.2700	0.1010	0.0356	0,0813	0.0300
pmed15	300	100	1729	3.5319	0.4600	0.4048	0.6106	0,0795	0.0700
pmed16	400	5	8162	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed17	400	10	6999	0.0000	0.0600	0.0000	0.0000	0,0419	0.0000
pmed18	400	40	4809	0.0402	0.1200	0.0415	0.0193	0,0577	0.0200
pmed19	400	80	2845	0.3854	0.1100	0.4920	0.1096	0,0312	0.1400
pmed20	400	133	1789	4.1121	0.1100	0.8384	0.7356	0,0196	0.1500
pmed21	500	5	9138	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed22	500	10	8579	0.0015	0.0000	0.0000	0.0013	0.0000	0.0000
pmed23	500	50	4619	0.0144	0.0000	0.0000	0.0066	0.0000	0.0000
pmed24	500	100	2961	0.3793	0.0300	0.2026	0.1123	0,0088	0.0600
pmed25	500	167	1828	4.5751	1.2000	0.9299	0.8363	0,2193	0.1700
pmed26	600	5	9917	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Conjunto de datos				FMEDIAN RPE			Desviación estándar		
Problema	n	P	Óptimo	HSOS	ABC	TS	HSOS	ABC	TS
pmed27	600	10	8307	0.0040	0.0000	0.0000	0.0033	0.0000	0.0000
pmed28	600	60	4498	0.1052	0.0200	0.1333	0.0473	0.0909	0.0600
pmed29	600	120	3033	0.2274	0.3000	0.1978	0.0690	0,0089	0.0600
pmed30	600	200	1989	4.8516	0.8000	1.2066	0.9650	0,1591	0.2400
			Promedio	0.7348	**0.1450**	0.2251	0.1359	**0.0309**	0.0553

El algoritmo SBHS demostró tener un buen resultado igual al óptimo (ver Tabla 28), en los problemas: Pmed1, Pmed3, Pmed6, Pmed11, Pmed12, Pmed21 y Pmed22; así mismo demostró obtener un mejor resultado que el algoritmo TS en el conjunto de datos Pmed2, Pmed4, Pmed7; así como también un mejor resultado al algoritmo ABC en el conjunto de datos: Pmed17. Y logro acercarse al óptimo de los problemas: Pmed16, Pmed26 y Pmed 29.

Tabla 28. Resultados SBHS

Conjunto de datos				FMEDIAN RPE			Desviación estándar		
Problema	n	P	Óptimo	SBHS	ABC	TS	SBHS	ABC	TS
pmed1	100	5	5819	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed2	100	10	4093	0.0586	0.0000	0.2931	0.0240	0.0000	0.1200
pmed3	100	10	4250	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed4	100	20	3034	0.1680	0.0000	0.3955	0.0510	0.0000	0.1200
pmed5	100	33	1355	3.0405	0.0000	0.0000	0.4120	0.0000	0.0000
pmed6	200	5	7824	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed7	200	10	5631	0.0177	0.0000	0.1420	0.0100	0.0000	0.0800
pmed8	200	20	4445	0.2557	0.0000	0.2024	0.1136	0.0000	0.0900
pmed9	200	40	2734	2.8627	0.1500	0.6949	0.7826	0.0000	0.1900
pmed10	200	67	1255	12.5816	0.7200	0.4780	1.5790	0.0410	0.0600
pmed11	300	5	7696	0.0000	0.0000	0.0000	0.0000	0,0903	0.0000
pmed12	300	10	6634	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed13	300	30	4374	1.2437	0.0000	0.0000	0.5440	0.0000	0.0000
pmed14	300	60	2968	5.5323	0.2700	0.1010	1.6420	0.0000	0.0300
pmed15	300	100	1729	13.3834	0.4600	0.4048	2.3140	0,0813	0.0700
pmed16	400	5	8162	0.0171	0.0000	0.0000	0.0140	0,0795	0.0000
pmed17	400	10	6999	0.0085	0.0600	0.0000	0.0060	0.0000	0.0000
pmed18	400	40	4809	1.8749	0.1200	0.0415	0.9016	0,0419	0.0200
pmed19	400	80	2845	6.5178	0.1100	0.4920	1.8543	0,0577	0.1400
pmed20	400	133	1789	17.5796	0.1100	0.8384	3.1450	0,0312	0.1500
pmed21	500	5	9138	0.0000	0.0000	0.0000	0.0000	0,0196	0.0000
pmed22	500	10	8579	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
pmed23	500	50	4619	2.9292	0.0000	0.0000	1.3530	0.0000	0.0000
pmed24	500	100	2961	7.8318	0.0300	0.2026	2.3190	0.0000	0.0600
pmed25	500	167	1828	18.0306	1.2000	0.9299	3.2960	0,0088	0.1700
pmed26	600	5	9917	0.0070	0.0000	0.0000	0.0070	0,2193	0.0000

Conjunto de datos				**FMEDIAN RPE**			**Desviación estándar**		
Problema	n	P	Óptimo	SBHS	ABC	TS	SBHS	ABC	TS
pmed27	600	10	8307	0.0024	0.0000	0.0000	0.0020	0.0000	0.0000
pmed28	600	60	4498	3.1021	0.0200	0.1333	1.3953	0.0000	0.0600
pmed29	600	120	3033	8.6130	0.3000	0.1978	2.6123	0.0909	0.0600
pmed30	600	200	1989	18.5034	0.8000	1.2066	3.6803	0,0089	0.2400
			Promedio	4.1387	**0.1450**	0.2251	0.9352	**0.0309**	0.0553

La clasificación de los mejores resultados obtenidos para el conjunto de datos OR-Library al hacer uso de los algoritmos HSOS y SBHS se muestra en la Tabla 29.

Tabla 29. Grupos de mejores resultados obtenidos

	HSOS	SBHS
Resultados	Conjunto de datos	Conjunto de datos
Encuentra el óptimo	Pmed1, Pmed3, Pmed6, Pmed11, Pmed12, Pmed13, Pmed21, Pmed22	Pmed1, Pmed3, Pmed6, Pmed11, Pmed12, Pmed21, Pmed22
Mejor que TS	Pmed2, Pmed4, Pmed8, Pmed9, Pmed17, Pmed18, Pmed19, Pmed28	Pmed2, Pmed4, Pmed7
Mejor que ABC	Pmed17, Pmed18	Pmed17
Cerca al óptimo	Pmed7, Pmed14, Pmed22, Pmed23, Pmed27, Pmed29	Pmed16, Pmed26, Pmed29

De acuerdo con las tablas presentadas, se puede observar que el algoritmo HSOS presenta mejores resultados que los obtenidos en SBHS, esto es debido a que el algoritmo hace uso de diferentes fases que permiten a una solución evolucionar dando una mayor diversificación en la población, mientras que el algoritmo SBHS genera una solución realizando una simplificación en los operadores habituales de búsqueda armónica, haciendo que la población quede con un número de instalaciones muy reducida para posteriormente ser reparada.

Ya que en los algoritmos presentados en el estado del arte no indican el número de evaluaciones a realizar, este proyecto tomo únicamente 1500 evaluaciones de acuerdo con el número de pruebas realizadas.

También se encontró que el algoritmo SBHS encuentra buenos resultados para los conjuntos de datos donde el número P de instalaciones a ubicar es menor a 20 y el máximo de vértices presentado para cada problema es menor o igual a 100, y para vértices mayores de 100, cuando el número de instalaciones a ubicar es menor o igual a 10.

El algoritmo HSOS encuentra el óptimo para los problemas que tienen cualquier número de vértices en donde el número P de instalaciones a ubicar es menor o igual a 30, y cuando se desea ubicar un número P de instalaciones mayor a 30, obtiene resultados cerca al optimo siempre y cuando no supere un número de 60 instalaciones.

Capítulo 6

6 CONCLUSIONES Y TRABAJO FUTURO

6.1 CONCLUSIONES

Este trabajo presento una adaptación de los algoritmos de HSOS y SBHS para el problema de la mochila en el contexto del problema de ubicación de instalaciones *P*_mediana no capacitada, y de mejora por medio de un algoritmo de búsqueda local.

La ejecución de estos algoritmos requiere de altas capacidades de cómputo por la complejidad de los problemas FLP, debido a la dimensionalidad y el número de medianas (*P*) a ubicar, por esto, en este proyecto se propone una arquitectura distribuida para mejorar los tiempos y alcanzar los resultados necesarios de procesamiento para este problema.

En la adaptación implementada de cada algoritmo, permite que cualquier algoritmo de búsqueda armónica o metaheuristica que trabaje con soluciones binarias, sean adaptadas al problema de ubicación de instalaciones P mediana no capacitado.

En la evaluación de los algoritmos se encontró con respecto al error relativo, que para HSOS se presentaron mejores resultados que los algoritmos del estado del arte en algunas instancias (12 problemas), debido a que este algoritmo tiene más fases durante la evolución que permiten diversificar las soluciones y luego reparar con conocimiento, y por ende se adapta mejor que el algoritmo SBHS debido a que este hace solo uso de una fase.

Con respecto a la desviación estándar, HSOS también presentó mejores resultados en las mismas instancias que para el error relativo, debido a que este algoritmo evoluciona haciendo un mayor número de fases antes de obtener una solución, haciendo que la diferencia entre los resultados sea más aproximada que las obtenidas en SBHS.

El algoritmo HSOS encuentra el óptimo para los problemas que tienen cualquier número de vértices en donde el número P de instalaciones a ubicar es menor o igual a 30, y obtiene resultados cerca al optimo cuando se desea ubicar un número P de instalaciones mayor a 30, pero menor a 60 instalaciones.

6.2 TRABAJO FUTURO

Incluir otros algoritmos de búsqueda local como el algoritmo de intercambio, además del método de adición y eliminación encontrados en el estado del arte que se adapten para encontrar mejores resultados en los algoritmos adaptados.

Incluir otros métodos de inicialización de la población que se ajusten a los algoritmos propuesto, esto es debido a que la reparación utilizada en la inicialización hace que el número de instalaciones sea pequeño.

Utilizar el conjunto de datos de Galvao [5] para problemas de menor complejidad que los encontrados en ORLIB con los algoritmos implementados, esto es debido a que todos los algoritmos obtuvieron buenos resultados en este problema para soluciones con pocos vértices.

Realizar más variación en los valores de los parámetros de los algoritmos adaptados, ya que por cuestiones de tiempo no fue posible hacerlo.

Ejecutar los algoritmos implementados con un número mayor de evaluaciones, esto es debido a que no se sabe el número exacto de evaluaciones aplicadas en el estado del arte.

Referencias bibliográficas

[1] G. Erdoğan, N. Stylianou, and C. Vasilakis, “An open source decision support system for facility location analysis,” *Decis. Support Syst.*, vol. 125, no. July, p. 113116, 2019, doi: 10.1016/j.dss.2019.113116.

[2] D. Celik Turkoglu and M. Erol Genevois, *A comparative survey of service facility location problems*, no. 0123456789. Springer US, 2019.

[3] S. Y. He, Y. H. Kuo, and D. Wu, “Incorporating institutional and spatial factors in the selection of the optimal locations of public electric vehicle charging facilities: A case study of Beijing, China,” *Transp. Res. Part C Emerg. Technol.*, vol. 67, pp. 131–148, 2016, doi: 10.1016/j.trc.2016.02.003.

[4] S. Mete, Z. A. Cil, and E. Özceylan, “Location and coverage analysis of bike-sharing stations in university campus,” *Bus. Syst. Res.*, vol. 9, no. 2, pp. 80–95, 2018, doi: 10.2478/bsrj-2018-0021.

[5] M. Basti and M. Sevkli, “An artificial bee colony algorithm for the p-median facility location problem,” *Int. J. Metaheuristics*, vol. 4, no. 1, p. 91, 2015, doi: 10.1504/ijmheur.2015.071769.

[6] T. Zhang and Z. W. Geem, “Review of harmony search with respect to algorithm structure,” *Swarm Evol. Comput.*, vol. 48, no. April 2018, pp. 31–43, 2019, doi: 10.1016/j.swevo.2019.03.012.

[7] N. Zarrinpoor, “An exploration of evolutionary algorithms for a bi-objective competitive facility location problem in congested systems,” *Int. J. Supply Oper. Manag.*, vol. 5, no. 3, pp. 266–282, 2018, doi: 10.22034/2018.3.6.

[8] Y. Oh, U. Park, and S. Kang, “Harmony search algorithm for high-demand facility locations considering traffic congestion and greenhouse gas emission,” in *Advances in Intelligent Systems and Computing*, 2016, vol. 382, pp. 317–327, doi: 10.1007/978-3-662-47926-1_31.

[9] I. Landa-Torres, J. Del Ser, S. Salcedo-Sanz, S. Gil-Lopez, J. A. Portilla-Figueras, and O. Alonso-Garrido, “A comparative study of two hybrid grouping evolutionary techniques for the capacitated P-median problem,” *Comput. Oper. Res.*, vol. 39, no. 9, pp. 2214–2222, 2012, doi: 10.1016/j.cor.2011.11.004.

[10] A. Kaveh and H. Nasr Esfahani, “Hybrid harmony search for conditional p-median problems,” *Int. J. Civ. Eng.*, vol. 10, no. 1, pp. 32–36, 2012.

[11] M. El-Shafei, I. Ahmad, and M. G. Alfailakawi, “Hardware accelerator for solving 0–1 knapsack problems using binary harmony search,” *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 33, no. 1, pp. 87–102, 2018, doi: 10.1080/17445760.2017.1324025.

[12] X. Kong, L. Gao, H. Ouyang, and S. Li, “A simplified binary harmony search algorithm for large scale 0-1 knapsack problems,” *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5337–5355, 2015, doi: 10.1016/j.eswa.2015.02.015.

[13] H. Wu, Y. Zhou, and Q. Luo, “Hybrid symbiotic organisms search algorithm for solving 0-1 knapsack problem,” *Int. J. Bio-Inspired Comput.*, vol. 12, no. 1, pp. 23–53, 2018, doi: 10.1504/ijbic.2018.093334.

[14] S. Tafazzoli and M. Mozafari, “Classification of location models and location softwares,” in *Contributions to Management Science*, Springer, 2009, pp. 505–521.

[15] B. Kitchenham and P. Brereton, “A systematic review of systematic review process research in software engineering,” *Inf. Softw. Technol.*, vol. 55, no. 12, pp. 2049–2075, 2013, doi: 10.1016/j.infsof.2013.07.010.

[16] V. Medvid, “Two efficient algorithms for weighted p-median problem,” *Commun. - Sci. Lett. Univ. Zilina*, vol. 17, no. 2, pp. 55–59, 2015.

[17] K. Jörnsten and A. Klose, “An improved Lagrangian relaxation and dual ascent approach to facility location problems,” *Comput. Manag. Sci.*, vol. 13, no. 3, pp. 317–348, 2016, doi: 10.1007/s10287-015-0244-z.

[18] R. Madleňák, L. Madleňáková, J. Štefunko, and R. Keil, “Multiple Approaches of Solving Allocation Problems on Postal Transportation Network in Conditions of Large Countries,” *Transp. Telecommun.*, vol. 17, no. 3, pp. 222–230, 2016, doi: 10.1515/ttj-2016-0020.

[19] M. Karatas and E. Yakıcı, “An iterative solution approach to a multi-objective facility location problem,” *Appl. Soft Comput. J.*, vol. 62, pp. 272–287, 2018, doi: 10.1016/j.asoc.2017.10.035.

[20] T. V. Levanova and A. Y. Gnusarev, “Simulated Annealing for Competitive p-Median Facility Location Problem,” *J. Phys. Conf. Ser.*, vol. 1050, no. 1, pp. 0–5, 2018, doi: 10.1088/1742-6596/1050/1/012044.

[21] T. Levanova and A. Gnusarev, *Development of Ant Colony Optimization Algorithm for Competitive p-Median Facility Location Problem with Elastic Demand*, vol. 1090 CCIS. Springer International Publishing, 2019.

[22] B. Afify, S. Ray, A. Soeanu, A. Awasthi, M. Debbabi, and M. Allouche, “Evolutionary learning algorithm for reliable facility location under disruption,” *Expert Syst. Appl.*, vol. 115, pp. 223–244, 2019, doi: 10.1016/j.eswa.2018.07.045.

[23] R. L. Church and C. A. Baez, “Generating optimal and near-optimal solutions to facility location problems,” *Environ. Plan. B Urban Anal. City Sci.*, vol. 47, no. 6, pp. 1014–1030, 2020, doi: 10.1177/2399808320930241.

[24] J. Chang, L. Wang, J. K. Hao, and Y. Wang, “Parallel iterative solution-based tabu search for the obnoxious p-median problem,” *Comput. Oper. Res.*, vol. 127, p. 105155, 2021, doi: 10.1016/j.cor.2020.105155.

[25] H. Gwalani, C. Tiwari, and A. R. Mikler, “Evaluation of heuristics for the p-median

problem: Scale and spatial demand distribution," *Comput. Environ. Urban Syst.*, vol. 88, no. August 2020, p. 101656, 2021, doi: 10.1016/j.compenvurbsys.2021.101656.

[26] O. Alp, E. Erkut, and Z. Drezner, "An Efficient Genetic Algorithm for the p-Median Problem," *Ann. Oper. Res.*, vol. 122, no. 1–4, pp. 21–42, 2003, doi: 10.1023/A:1026130003508.

[27] E. Rolland, D. A. Schilling, and J. R. Current, "An efficient tabu search procedure for the p-Median Problem," *Eur. J. Oper. Res.*, vol. 96, no. 2, pp. 329–342, 1997, doi: 10.1016/S0377-2217(96)00141-5.

[28] F. Chiyoshi and R. D. Galvão, "A statistical analysis of simulated annealing applied to the p-median problem," *Ann. Oper. Res.*, vol. 96, no. 1–4, pp. 61–74, 2000, doi: 10.1023/a:1018982914742.

[29] C.-C. Lin and Y.-I. Chiang, "Alternative formulations for the obnoxious p-median problem," *Discret. Appl. Math.*, vol. 289, pp. 366–373, 2020, doi: 10.1016/j.dam.2020.11.002.

[30] E. Baş and E. Ülker, "A binary social spider algorithm for uncapacitated facility location problem," *Expert Syst. Appl.*, vol. 161, 2020, doi: 10.1016/j.eswa.2020.113618.

[31] K. Pratt, "Design Patterns for Research Methods: Iterative Field Research," *AAAI Spring Symp. Exp. Des. Real*, no. 1994, pp. 1–7, 2009.