Metaheurístico basado en *multi-space sampling* para la solución de problemas de distritación

Carlos Alberto Sánchez Alzate

Trabajo de grado presentado para optar al título de Ingeniero de Sistemas

Asesor

Juan G. Villegas Ph.D. in Systems Optimization

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Antioquia, Colombia
2022

Dedicatoria

A mis padres, quienes han guiado mi camino con sabiduría.

Tabla de contenido

Lista de tablas

Lista de figuras

Siglas, acrónimos y abreviaturas

APA	American Psychological Association
Cms.	Centímetros
ERIC	Education Resources Information Center
Esp.	Especialista
MP	Magistrado Ponente
MSc	Magister Scientiae
Párr.	Párrafo
PhD	Philosophiae Doctor
PBQ-SF	Personality Belief Questionnaire Short Form
PostDoc	PostDoctor
UdeA	Universidad de Antioquia

Resumen

Los problemas de distritación (PD) buscan agrupar pequeñas áreas geográficas (clientes, manzanas, barrios, etc.), en grupos geográficos de mayor tamaño de manera que estos últimos sean útiles según las características del problema. Las aplicaciones de este problema comprenden desde la zonificación política, diseño de zonas de ventas y distribución, hasta la planeación de cuidado domiciliario, entre otras. Para la solución de los PD se han utilizado en el pasado distintas técnicas, que van desde heurísticos constructivos y de búsqueda local hasta metaheurísticos, y métodos exactos basados en programación entera. En el presente trabajo se desarrolló una solución metaheurística para el problema de distritación general que podría ser aplicado a cualquier necesidad de distritación. La solución implementada está basada en un nuevo método metaheurístico llamado *multi-space sampling heuristic* (MSH) introducido por Mendoza & Villegas (2013). Este informe ilustra la revisión de la literatura asociada a los problemas de distritación y sus soluciones, la explicación realizada del MSH propuesto, los resultados computacionales al aplicar la solución a un problema particular y el análisis de los resultados obtenidos.

Palabras clave: multi-space sampling, heurístico, msh, distritación, optimización

Abstract

Districting problems (DP) seek to group small geographic areas (customers, blocks, neighborhoods, etc.) into larger geographic groups so that the latter are useful according to the characteristics of the problem. The applications of this problem range from political zoning, to the design of sales and distribution areas, to home healthcare planning, among others. Different techniques have been used in the past to solve DPs, ranging from constructive and local search heuristics to metaheuristics and exact methods based on integer programming. In the present work, a metaheuristic solution for the general districting problem was developed that could be applied to any districting application. The implemented solution is based on a new metaheuristic framework called the multi-space sampling heuristic (MSH), introduced by Mendoza & Villegas (2013). This report illustrates the review of the literature associated with districting problems and their solutions; the explanation of the proposed MSH carried out; the computational results when applying the solution to a particular problem; and the analysis of the results obtained.

*Keywords***:** the multi-space sampling heuristic, msh, districting problems, optimization

Introducción

Los problemas de distritación o zonificación (PD) buscan agrupar pequeñas áreas geográficas (clientes, manzanas, barrios, etc.), llamadas unidades básicas, en grupos geográficos más grandes, llamados distritos, de manera que estos últimos sean aceptables según los criterios de planificación relevantes (Kalcsics, 2015). Aplicaciones de estos problemas aparecen en un sinnúmero de áreas tales como zonificación política (Bozkaya et al, 2003), diseño de zonas de ventas y distribución (Ríos-Mercado, & Escalante, 2016), planeación de cuidado domiciliario (Cortes et al 2018), entre otras. Encontrar una solución ingenieril a este tipo de problemas podría generar ahorros en el proceso de distribución de un producto, o el servicio de los pacientes hospitalizados en casa en la atención médica domiciliaria , o mejorar la atención que brindan las entidades gubernamentales a sus usuarios. Estos ejemplos permiten concluir que generar una solución al problema PD, es de especial relevancia en muchos sectores económicos.

El PD es un problema complejo debido a las condiciones para modelar el sistema, los requerimientos para prestar el servicio, y la complejidad computacional para resolverlo. Según la revisión de la literatura, es difícil encontrar soluciones óptimas e incluso factibles utilizando formulaciones matemáticas cuando el tamaño de la instancia aumenta por encima de las 50 unidades básicas o áreas que conforman el territorio. Debido a lo anterior, surge la necesidad de recurrir a enfoques aproximados tales como los heurísticos constructivos y de búsqueda local hasta metaheurísticos, y métodos exactos basados en programación entera (Ríos-Mercado, 2019).

Por otro lado, los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y la mecánica estadística" (Pinter et al , 1999).

Las metaheurísticas generalmente se aplican a problemas que no tienen un algoritmo o heurístico específico que dé una solución satisfactoria; o bien cuando no es posible implementar un método de solución exacto. La mayoría de las metaheurísticos tienen como objetivo los

problemas de optimización combinatoria, pero por supuesto, se pueden aplicar a cualquier problema que se pueda reformular en términos heurísticos.

Este trabajo describe el desarrollo de una solución metaheurística para el problema de distritación general que podría ser aplicado a cualquier aplicación de distritación en particular.

Como resultado de este trabajo de grado se implementó un nuevo método metaheurístico de solución para problemas de distritación basado en los métodos de *multi-space sampling heuristic* (MSH) introducidos por Mendoza & Villegas (2013). Para evaluar el desempeño del método se realizó un análisis de sus resultados al variar diferentes parámetros del algoritmo en instancias de la literatura del contexto de atención médica domiciliaria de la ciudad de Medellín.

Este documento tiene la siguiente estructura, inicialmente se presenta una revisión de la literatura que resume las diversas soluciones que a lo largo del tiempo se le ha dado al PD bajo diversas condiciones. Posteriormente, se describe el marco teórico que da sustento al método de solución propuesto. A continuación se presenta el algoritmo MSH desarrollado y sus componentes. Finalmente se presentan los resultados obtenidos en la experimentación computacional y las conclusiones.

1 Planteamiento del problema e hipótesis

Los problemas de distritación (PD) buscan agrupar pequeñas áreas geográficas (clientes, manzanas, barrios, etc.), llamadas unidades básicas, en grupos geográficos más grandes, llamados distritos, de manera que estos últimos sean aceptables según los criterios de planificación relevantes (Kalcsics, 2015). Por su parte, los problemas de enrutamiento de vehículos (PEV) buscan diseñar rutas de atención para clientes dispersos geográficamente utilizando una flota de vehículos que se encuentran ubicados en un centro de distribución u operación de manera que la distancia total recorrida por los vehículos sea mínima. De igual manera, para resolver los PEV se utilizan técnicas heurísticas, metaheurísticas y exactas basadas en programación entera (Toth & Vigo,2014). Los problemas de distritación y los problemas de enrutamiento de vehículos guardan cierta similitud en tanto que en ambos casos es necesario particionar un conjunto de elementos (unidades básicas o clientes, respectivamente) en grupos más grandes (distritos o rutas respectivamente) de manera que se cumplan algunas restricciones (conexidad de los distritos o capacidad de los vehículos respectivamente) y se optimicen algunos criterios (en ambos casos comúnmente relacionados con la distancia entre las unidades básicas o clientes).

Por esta razón, en este trabajo se buscará adaptar los métodos basados en MSH (Mendoza & Villegas, 2013, Montoya et al, 2016) para la solución de PDs. Como resultado se espera desarrollar un nuevo método metaheurístico de solución para problemas de distritación.

2 Objetivos

2.1 Objetivo general

Desarrollar un algoritmo metaheurístico basado en multi-space sampling para solución del problema de zonificación.

2.2 Objetivos específicos

● Adaptar conceptualmente los componentes del metaheurístico MSH para la solución de problemas de distritación

● Implementar computacionalmente el metaheurístico MSH para distritación usando Python como lenguaje de programación

● Evaluar el desempeño del metaheurístico MSH para la solución de problemas de distritación usando instancias de prueba de la literatura.

3 Revisión de la Literatura

El problema de distritación se ha estudiado en un sin número de oportunidades por diferentes autores y con diversos enfoques; a continuación se mencionan los avances más relevantes.

Hess (1971) desarrolló un método exacto para dar solución específica al problema de distritación, posteriormente C. Easingwood (1973) y Lodish (1975) propusieron heurísticos para dar solución al mismo problema. Años más tarde, otros autores han desarrollado otros métodos, por ejemplo, Fleischmann (1988) desarrolló una solución para optimizar la asignación de vendedores a las zonas de interés, Ferland (1990) proporcionó una solución para las escuelas, Bergey (2003) proporcionó una solución para el sector eléctrico, Caro (2004) abordó la optimización de la educación, Ríos-Mercado (2009) proporcionó una solución para asignación de territorios de ventas.

El problema de distritación fue abordado por Blais et al (2003) en el cual se planteó una solución para la optimización de un servicio de salud teniendo un equilibrio de la carga de trabajo de cada distrito y respetando la contigüidad de los distritos.

Especial atención deben tener las soluciones para optimización de ventas, ya que son las que tienen mayor número de publicaciones. La necesidad consiste en agrupar unidades de ventas en distritos para asignar cada distrito a un representante de ventas. En este caso es importante que los distritos seleccionados tengan un tamaño similar en términos del número de clientes o de carga de trabajo o de potencial de ventas. En caso de que se haga una distribución aleatoria (sin considerar los factores antes mencionados), a algunos vendedores les serían asignados distritos con bajo potencial de ventas y eso puede llevar a la desmotivación de la fuerza de ventas. Hess y Samuels (1971) fueron los primeros en proporcionar una solución para este problema utilizando un modelo de asignación-ubicación para maximizar la compacidad total de todos los distritos mientras se balancea la “actividad” de los vendedores. Fleishman y Paraschis (1988) plantearon un modelo de asignación de territorios de ventas basado en asignación-ubicación que respeta la carga de trabajo,

la compacidad de los distritos y la indivisibilidad de las unidades básicas. Zoltners y Sinha (2005) plantearon un modelo de optimización multiobjetivo. Rios-Mercado y Fernandez (2009) sugirieron un modelo asignación-ubicación donde el objetivo es la maximización de la compacidad total mientras se balancea la actividad entre distritos contigüos.

Algunos autores como Skiera & Albers (1994) y Drexl & Haase (1999) argumentan que el objetivo de un modelo no debería ser lograr un balance entre distritos, puesto que la principal meta de una empresa es ganar dinero y por tanto plantearon un modelo que busca maximizar las ganancias.

Otro de los problemas comunes abordados en la literatura es la división administrativa de territorios para la gestión pública. En este caso se busca encontrar una agrupación óptima de calles, ciudades, barrios, estados, etc. con el fin de lograr un buen nivel de atención por parte de algún prestador de servicios públicos o hacer una asignación de distritos electorales u optimizar alguna variable de interés. “En muchos países se tiene la costumbre de redefinir los distritos electorales cada cierto tiempo, por ejemplo en Nueva Zelanda se hace cada 5 años, en Canadá y Estados Unidos cada 10 años” (Kalcsics, 2015). Tener un buen modelo de distritación es de gran ayuda para esta labor.

Otros dos criterios importantes que siempre se mencionan en este problema son la contigüidad y la compacidad. Mientras que la contigüidad es generalmente indiscutible y fácil para verificar, este no es el caso de la compacidad. Hay una amplia discusión sobre cómo cuantificar adecuadamente este criterio (Horn et al. 1993), y si es pertinente en primer lugar porque un algoritmo nunca se manipulará a propósito debido a que no usa datos políticos (Garfinkel y Nemhauser 1970). Para una revisión actualizada del problema de distritación se puede recurrir a Ríos Mercado (2020)

Por otro lado, *Multi-space Sampling Heuristic* (MSH) es un metaheurístico introducido por Mendoza & Villegas (2013) inicialmente planteado para dar solución al problema de enrutamiento de vehículos con demanda estocástica. Posteriormente, Montoya et al (2016) lo implementaron en el enrutamiento de vehículos propulsados por energías alternativas para optimizar sus rutas

teniendo en cuenta la disponibilidad de estaciones de recarga y la limitada capacidad de sus reservorios de energía. Su principio básico consiste en ordenar primero los clientes que se requiere visitar usando para ello un heurístico constructivo aleatorizado, el cual se emplea para alimentar un conjunto de columnas factibles. Las cuales se usan para formular un problema de partición de conjuntos que se resuelve con un modelo de optimización (típicamente haciendo uso de un optimizador comercial). MSH ha sido utilizado con éxito en la solución de otros problemas de enrutamiento complejos: con demanda y tiempos estocásticos o con estructuras complejas en sus rutas . Sin embargo, no ha sido utilizado aún para la solución de otros problemas de optimización combinatoria; el problema de distritación es uno de ellos.

4 Marco Teórico

4.1 Conceptos Importantes

Unidades básicas (BU): Una unidad básica es aquella que representa el objeto a ser asignado. También es llamado: unidad de cobertura de ventas, área básica o unidad geográfica. La definición específica depende del contexto del problema a resolver, en algunos casos puede ser una figura geométrica, un punto, un segmento de línea, una calle o un área poligonal (código de un área postal, un barrio, condado, etc.).

Distritos: Es un subconjunto de Unidades Básicas (BU). El número de distritos puede ser un número fijo y previamente conocido según el contexto del problema. Por ejemplo, en un problema de asignación de enfermeras para atender personas de la tercera edad en los barrios, el número de distritos estaría determinado por la cantidad de enfermeras disponibles; o en un problema de asignación de patrullas de la policía para la atención de los barrios, la cantidad de distritos estaría dado por la cantidad de patrullas disponibles. En algunos casos es importante determinar cuál será la mejor ubicación del centro de cada distrito por ejemplo para ubicar en ese punto al recurso que atiende a todos los pacientes del distrito cuando lo solicitan.

Existe un conjunto importante de criterios asociados a los problemas de distritación, los cuales se mencionan a continuación:

Balance: Describe el deseo de distritos de tamaño equitativo con respecto a alguna medida de desempeño para los distritos. Dependiendo del contexto, este criterio puede tener una motivación económica, por ejemplo, potencial de ventas, carga de trabajo o número de clientes equitativos, el mismo número de habitantes o votantes elegibles.

Contigüidad: Un distrito se llama contigüo si es posible viajar entre las unidades básicas del distrito sin tener que salir del distrito.

Compacidad: Finalmente, se dice que un distrito es geográficamente compacto si tiene una forma circular, no distorsionada y sin agujeros. Los distritos contiguos y compactos suelen reducir el tiempo de viaje de la persona responsable de dar servicio al distrito. (Kalcsics, 2015).

4.2 Formulación Matemática del Problema de Distritación

El problema de distritación (DP) puede ser descrito de la siguiente manera: La partición de todas las I unidades básicas en un número K de distritos que satisface los criterios de balance o compacidad y contigüidad, y si lo requiere que localice un centro dentro de cada distrito.

Para la formulación matemática se utilizó un modelo de Programación Entera Mixta (MIP). El modelo usado se denomina FLOW^{PRM} (FLOW p-regions model) o modelo de p-regiones con variables de flujo, a través de las cuáles se logra la restricción de contigüidad entre las UBs. Este trabajo usa la versión Flow del problema de p-regiones formulado por Duque, Church, & Middleton (2011).

La formulación, tomada de Cortes et al (2018) sería:

Parámetros:

$$i, I = índice\ y\ conjunto\ de\ UBs, i = \{1, \dots, n\};$$
$$k, K = índice\ y\ conjunto\ de\ distritos, k = \{1, \dots, p\};$$
$$N_i = \{c_{ij} = 1\}, conjunto\ de\ UBs\ que\ son\ adyacentes\ a\ la\ UB\ i;$$
$$c_i = carga\ de\ trabajo\ medida\ en\ horas\ de\ las\ UBs, i = \{1, \dots, n\};$$
$$d_{ij} = distancia\ entre\ la\ unidad\ básica\ i\ y\ la\ unidad\ básica\ j;$$

Las variables de decisión para este modelo son las siguientes:

$$t_{ij} = \{1, si\ las\ UBs\ i\ y\ j\ pertenecen\ al\ mismo\ distrito\ k\ con\ i < j\ ; 0, en\ otro\ caso;$$

$$f_{ijk} = variable\ continua\ no\ negativa\ que\ indica\ la\ cantidad\ de\ flujo\ de\ la\ UB\ i\ a\ la\ j\ en\ el\ distrito\ k$$

$$y_{ik} = \{1, si\ la\ UB\ i\ es\ incluida\ en\ el\ distrito\ k, 0\ en\ otro\ caso;$$

$$w_{ik} = \{1, si\ la\ UB\ i\ es\ elegida\ como\ un\ sumidero\ del\ distrito\ k, 0\ en\ otro\ caso;$$

$$CT_k = Carga\ de\ trabajo\ del\ distrito\ k;$$

La formulación matemática del DP en HHC basada en FLOW^{PRM} se presenta a continuación:

$$Min\ Z = Min\ Distancia\ \ Total = \sum_{j=1}^{I} \sum_{i=1}^{I} tij * dij \quad (1)$$

Sujeto a:

$$\sum_{k=1}^{p} y_{ik} = 1 \quad \forall i = 1, \dots, n, \quad (2)$$

$$w_{ik} \leq y_{ik} \quad \forall i = 1, \dots, n;\ \forall k = 1, \dots, p, \quad (3)$$

$$\sum_{i=1}^{n} w_{ik} = 1 \quad \forall k = 1, \dots, p, \quad (4)$$

$$f_{ijk} \leq y_{ik} \cdot (n - p) \quad \forall i = 1, \dots, n;\ \forall j \in N_i;\ \forall k = 1, \dots, p, \quad (5)$$

$$f_{ijk} \leq y_{jk} \cdot (n - p) \quad \forall i = 1, \dots, n;\ \forall j \in N_i;\ \forall k = 1, \dots, p, \quad (6)$$

$$\sum_{j \in N_i} f_{ijk} - \sum_{j \in N_i} f_{jik} \geq y_{ik} - (n - p) \cdot w_{ik} \quad \forall i = 1, \dots, n;\ \forall k = 1, \dots, p, \quad (7)$$

$$t_{ij} \geq y_{ik} + y_{jk} - 1 \quad \forall i, j = 1, \dots, n | i < j;\ \forall k = 1, \dots, p, \quad (8)$$

$$CT_k = \sum_{i \in I} c_i y_{ik};\ \forall k = 1, \dots, p. \tag{9}$$

$$y_{ik} \in \{0,1\} \quad \forall i = 1, \dots, n;\ \forall k = 1, \dots, p, \tag{10}$$

$$w_{ik} \in \{0,1\} \quad \forall i = 1, \dots, n;\ \forall k = 1, \dots, p, \tag{11}$$

$$t_{ij} \geq 0 \quad \forall i, j = 1, \dots, n | i < j, \tag{12}$$

$$f_{ijk} \geq 0 \quad \forall i = 1, \dots, n;\ \forall j \in N_i;\ \forall k = 1, \dots, p. \tag{13}$$

$$(1 + \alpha) * \frac{\sum_{i \in I} c_i}{|K|} \leq CT_k \leq (1 + \alpha) * \frac{\sum_{i \in I} c_i}{|K|} \qquad \forall k = 1, \dots, p. \tag{14}$$

5 Materiales y Métodos

5.1 Descripción del algoritmo multi-space sampling (MSH)

La idea detrás del MSH es muestrear diferentes representaciones de espacios de solución y luego ensamblar una solución con partes de los elementos muestreados.

El pseudocódigo del algoritmo está expresado en la **Figura 1**. Para comenzar, en las líneas 2 y 3 se inicializan las variables del problema, en este caso Ω corresponde a la biblioteca de distritos y *t* corresponde al contador de iteraciones. Para un mejor entendimiento del algoritmo se divide en dos fases. La primera fase denominada fase de muestreo, usa un heurístico *h* (comúnmente el de ordenar primero enrutar después), usado en forma aleatorizada para muestrear diferentes espacios de representación de soluciones obteniendo una permutación de los distritos p^t (línea 5). El muestreo se inicia en una unidad básica seleccionada aleatoriamente, y continúa seleccionando unidades básicas de la misma forma, teniendo en cuenta el criterio de adyacencia (en caso de que sea válido para el problema). Una vez construida la permutación p^t se extrae el subconjunto de distritos factibles Ω^t que se pueden construir a partir del orden de las unidades básicas obtenido en p^t. El procedimiento (feasible(Ω^t)) extrae únicamente los distritos que cumplen con los criterios relevantes para el problema: por ejemplo adyacencia, compacidad y/o límite máximo de la carga de trabajo (línea 6). Este proceso se repite durante una cantidad definida de iteraciones (T).

En la segunda fase denominada fase de ensamble, se genera una nueva solución a partir del conjunto Ω conformado por todos los distritos factibles únicos obtenidos en la fase de muestreo.

En términos generales, en la segunda fase, MSH, construye un conjunto Ω, siguiendo el principio del heurístico de los pétalos para enrutamiento de vehículos (Foster and Ryan, 1976) y mapea el conjunto Ω a una solución ($\bar{R}$) resolviendo una formulación de partición de conjuntos (línea 10).

En este caso, en la fase ensamble se encuentra la combinación de distritos factibles que optimiza la función objetivo, dicha función minimiza la distancia entre todas las unidades básicas que pertenecen a un mismo distrito (para ello suman las distancias entre los arcos en los cuales cada vértice corresponde a una unidad básica)

```
1: function MULTISPACESAMPLING (G,h,T)
2:      Ω ←0
3:      t ←1
4:      while t⩽T
5:           p^t ← h(G)
6:           Ω^t ← feasible(p^t)
7:           Ω ← ΩUΩ^t
8:           t ← t+1
9:      end while
10:     R̄ ← setPartitioning(G,Ω,)
11:     return R̄
12: end function
```

Figura 1. Pseudocódigo del Algoritmo MSH

5.2 Formulación del problema de partición

Para la fase de ensamble se usa el problema de partición de conjuntos usando la siguiente formulación:

Notación:

Parámetros:

a_{ik}: 1 si la BU i está en el distrito k, 0 si no está

d_k: Distancia total del distrito k

Variables:

x_k: Es 1 si el distrito k ∈ Ω es seleccionado dentro de la solución final, 0 de lo contrario.

$$Min\ Distancia\ \ Total = \sum_{k\ \in\ \Omega} \quad x_k\ * d_k$$

Restricciones:

Cantidad máxima de distritos:

$$\sum_{k\in\Omega} \quad x_k\ = K$$

Cada BU debe ser asignada a un único distrito:

$$\sum_{k\in\Omega} \quad x_k * a_{ik} = 1 \quad \forall i = 1, \dots, n$$

Dominio de las variables:

$$x_k = \{0,1\} \quad \forall\ k \in \Omega$$

5.3 Ejemplo de Multi-space Sampling

Para facilitar el entendimiento del algoritmo propuesto, a continuación se desarrolla un ejemplo. El enunciado del Problema para el ejemplo podría ser el siguiente, se tiene un conjunto de unidades básicas (A, B, … , I) con su respectiva carga de trabajo y graficadas de acuerdo a su adyacencia (si dos unidades básicas tienen un lado en común significa que son adyacentes). Ver **Figura 2**.

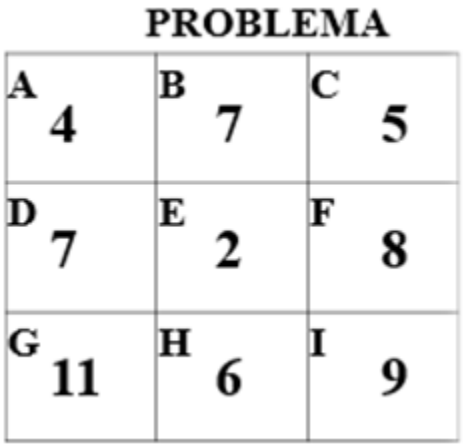

Figura 2. Esquema de un problema pequeño de distritación.

Se tiene un Desbalance máximo de 22.61, lo cual quiere decir que los distritos cuya carga supere ese valor deben ser descartados (distritos en rojo en las imágenes).

Se procede a realizar permutaciones aleatorizadas de Unidades Básicas teniendo en cuenta la adyacencia. Ver **Figura 3**.

Iteración: t=1 ,Permutación p^1 : ADGHIFEBC , Ω^1

Carga	4	11	22	28	7	18	24	...	12	5
Ubs	A	A	A	A	D	D	D		B	C
		D	D	D		G	G		C	
			G	G			H			
				H						

Iteración: t=2, Permutación p^2 :ADEBCFIHG, Ω^2

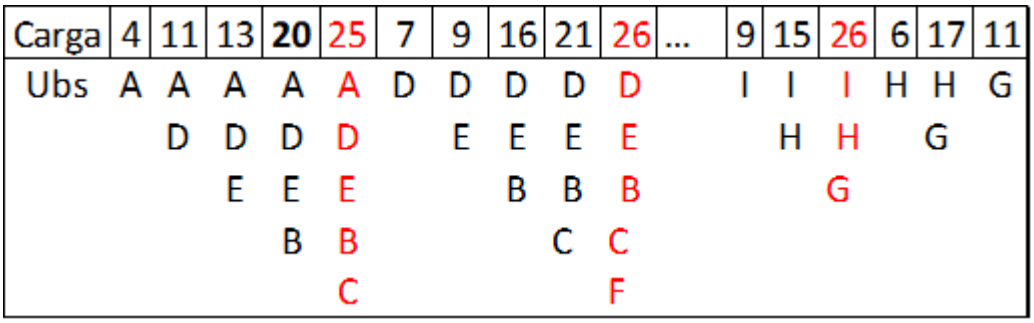

Figura 3. Ejemplos de permutaciones de unidades básicas

Se repite este procedimiento un número dado de veces (T iteraciones) y se guardan los distritos. Al final se obtiene una biblioteca de distritos factibles (Ω) descartando los distritos en rojo (los cuales no cumplen los criterios relevantes para el problema a resolver) porque en este caso en particular superan el desbalance máximo.

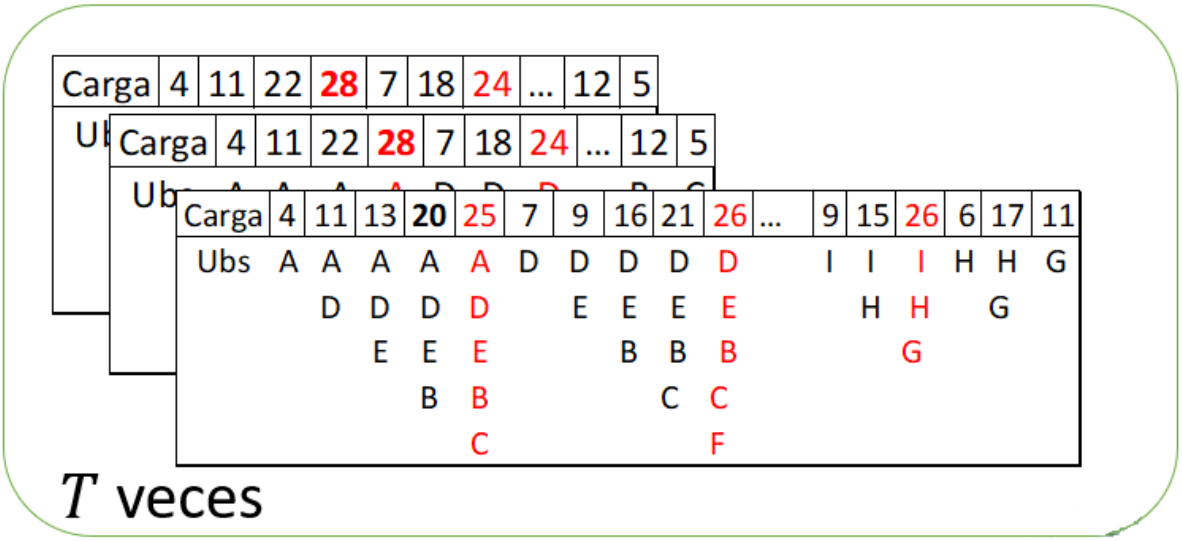

Figura 4. Ejemplos de varias iteraciones

Se procede a realizar la optimización de la función objetivo con la biblioteca de distritos (Ω) y a encontrar una solución al problema haciendo una partición sobre Ω.

6 Implementación de MSH

La implementación se llevó a cabo en Python siguiendo el pseudo código explicado anteriormente. En la **Figura 5** se muestran las etapas detalladas del algoritmo de implementación utilizando Python. Para la solución del modelo de optimización de partición de conjuntos de la fase de ensamble se utilizó el optimizador comercial Gurobi.

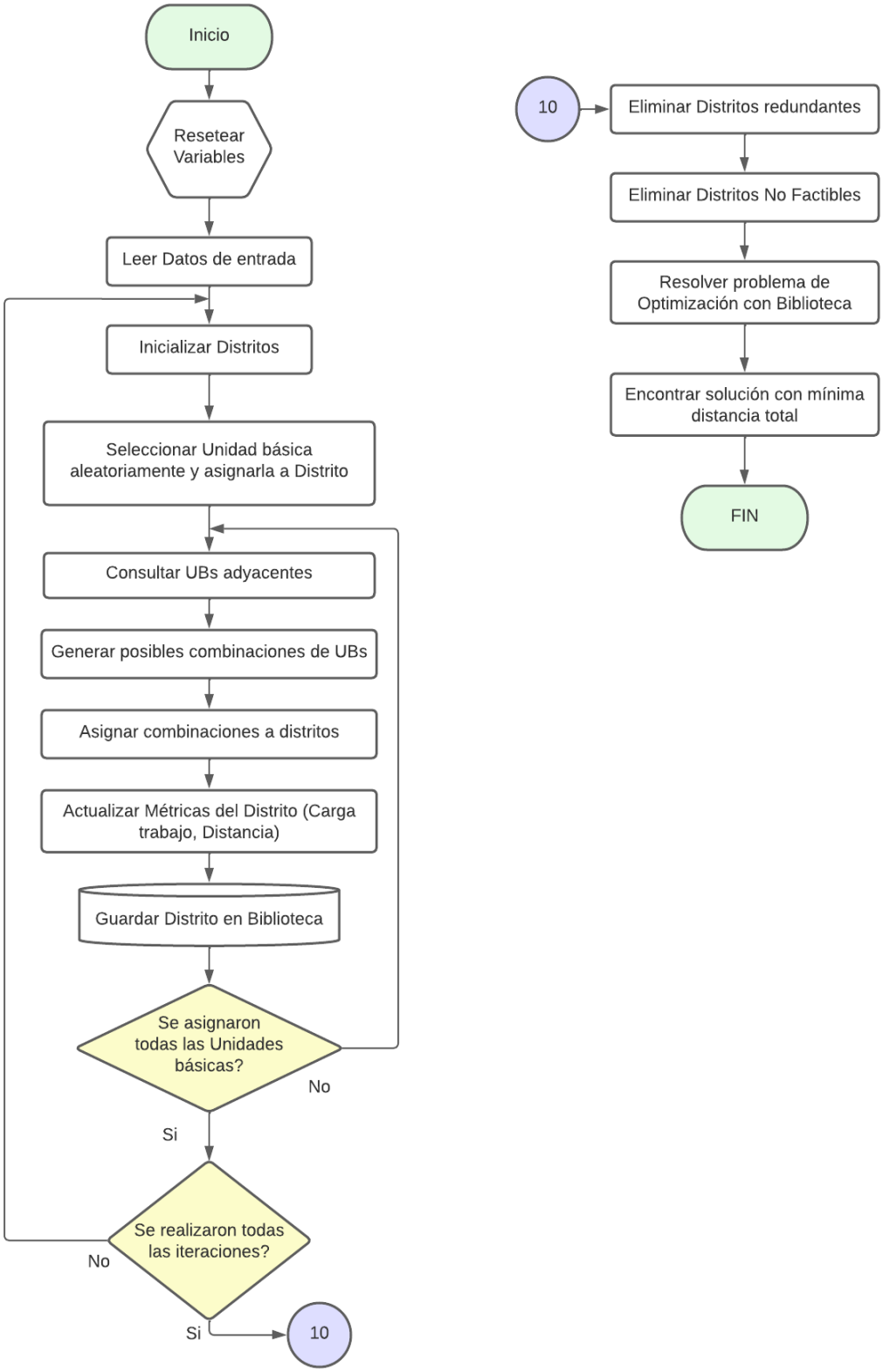

Figura 5. Diagrama de flujo de implementación de MSH.

7 Resultados y Discusión

En esta sección, se muestran los resultados de los experimentos llevados a cabo luego de la implementación del MSH para dar solución al problema de distritación.

La implementación se llevó a cabo en Python utilizando un computador con un Procesador Intel(R) Core(TM) i5- a 2.50GHz, con 2 procesadores principales, 4 procesadores lógicos bajo el Sistema Operativo Microsoft Windows 10 Pro x64 con una memoria física instalada (RAM) de 8.00 GB.

Para obtener los resultados de la optimización final se utilizó el optimizador comercial Gurobi versión 9.5.1.

Con el fin de comparar los resultados de este algoritmo con los obtenidos por otros autores, se tomaron las instancias de Cortes (2018), las cuales se ilustran en la **Tabla 1**.

Tabla 1.
Instancias consideradas para los experimentos computacionales

Instancia	UBs	Descripción
1	16	Comunas de la ciudad de Medellín
2	29	Barrios del municipio de Copacabana
3	44	Barrios del municipio de Envigado
4	60	Barrios del municipio de Itagüí
5	78	Barrios del municipio de Bello
6	104	Unir los Barrios de Envigado e Itagüí por ser vecinos
7	107	Unir los Barrios de Bello y Copacabana por ser vecinos
8	270	Barrios de la ciudad de Medellín

7.1 Determinación de distritos

La cantidad de distritos puede afectar la complejidad del sistema a resolver; en este caso, para fijar el número de distritos, se utilizó el mismo método usado por otros autores (Benzarti et al, 2013) y (Cortés et al, 2018), es decir, para cada tamaño de instancia se resolvió el problema dividiendo el territorio en 2, 3 y 4 distritos, y para ser proporcionales con el tamaño de las instancias, el número de distritos también se definió con los percentiles 25%, 50% y 75% del número de unidades básicas de cada instancia. Por ejemplo, para la instancia de 44 UBs, se dividió el territorio en 2, 3, 4, 11 (25%), 22 (50%) y 33 (75%) distritos.

Asi, mismo, se tiene dos versiones del heurístico MSH :

En la Versión 1, de cada una de las iteraciones se guardan los distritos y se almacenan en una biblioteca de distritos. El modelo de optimización se corre para cada iteración seleccionando la mejor solución, al final se corre el modelo teniendo en cuenta toda la biblioteca de distritos que fueron creados en cada iteración.

7.2 Resultados computacionales y análisis

Tal como lo ilustran los resultados en **Tabla 2**, como en esta versión el modelo de Set partitioning debe ejecutarse completamente en cada iteración, el tiempo de ejecución es bastante elevado, por tanto esta versión solo se probó para la instancia de 16 unidades básicas.

Tabla 2.
Resultados de los experimentos computacionales para la versión 1

Iteraciones		**1.000**		**2.000**	
Cantidad BUs	**Distritos**	**Distancia**	**Tiempo Ejecución (s)**	**Distancia**	**Tiempo Ejecución (s)**
16	2	232.49	804.9	232.49	2 437.8
	3	123.71	2 261.9	120.17	5 858.5
	4	67.95	2 884.4	67.95	6 263.5
	8	16.95	1 846.4	16.95	7 479.4

Versión 2: Se generan distritos aleatorios en cada iteración (sin correr el modelo de optimización), guardando todos los distritos en una biblioteca de distritos, al final de las T iteraciones se corre el

modelo de Set partitioning teniendo en cuenta la biblioteca de distritos y se obtiene la solución final. Este modelo se corrió para las instancias de 16, 29, 44, 60, 78, 104, 107.

Se ejecutó un experimento reducido con T=100 iteraciones. Como se puede ver en la **Figura 6**, la distancia de cada distrito se reduce a medida que se incrementa el tamaño de los distritos. En esta oportunidad se limitó la cantidad de iteraciones a 100 con el fin de obtener datos de las instancias más grandes en cantidad de unidades básicas y por tanto fue posible correr el modelo hasta la instancia de 107 unidades básicas. En la **Figura 7** se puede observar la evolución del tiempo de ejecución para cada instancia variando la cantidad de distritos. Los resultados se muestran en la **Tabla 3**.

Tabla 3.
Resultados de los experimentos computacionales para la versión 2 (T=100 iteraciones).

Cantidad BU	Distritos	Distancia (Km):	Tiempo Ejecución (s)
16	2	274.58	1.5
	3	141.78	1.6
	4	76.56	1.3
	8	16.95	2.1
	12	6.06	2.6
29	2	250.66	12.3
	3	148.85	12.9
	4	99.48	19.7
	7	30.69	12.8
	15	4.38	11.1

	22	1.58	8.6
44	2	804.06	37.5
	3	518.19	62.3
	4	362.48	79.3
	11	60.72	38.0
	22	10.19	43.0
	33	3.63	68.9
60	2	1 505.17	88.2
	3	995.73	118.3
	4	726.2	205.4
	15	71.53	105.6
	30	11.01	98.1
	45	3.22	100.1
78	2	3 383.65	184.3
	3	2 273.92	225.5
	4	1 623.72	607.4
	20	125.75	558.5
	39	15.10	236.9
	59	4.210	244.6
104	2	7 492.23	779.5
	3	4 810.49	1 269.9
	4	3 581.01	1 397.7
	26	188.48	2 355.2
	52	22.90	1 537.9
	78	6.49	2 606.5

107	2	9 835.91	618.7
	3	6 556.73	596.4
	4	4 893.20	1824.2
	27	204.71	873.0
	54	19.88	573.7
	80	6.11	616.8

Figura 6. Gráficas comparativas de Distancia para 100 iteraciones

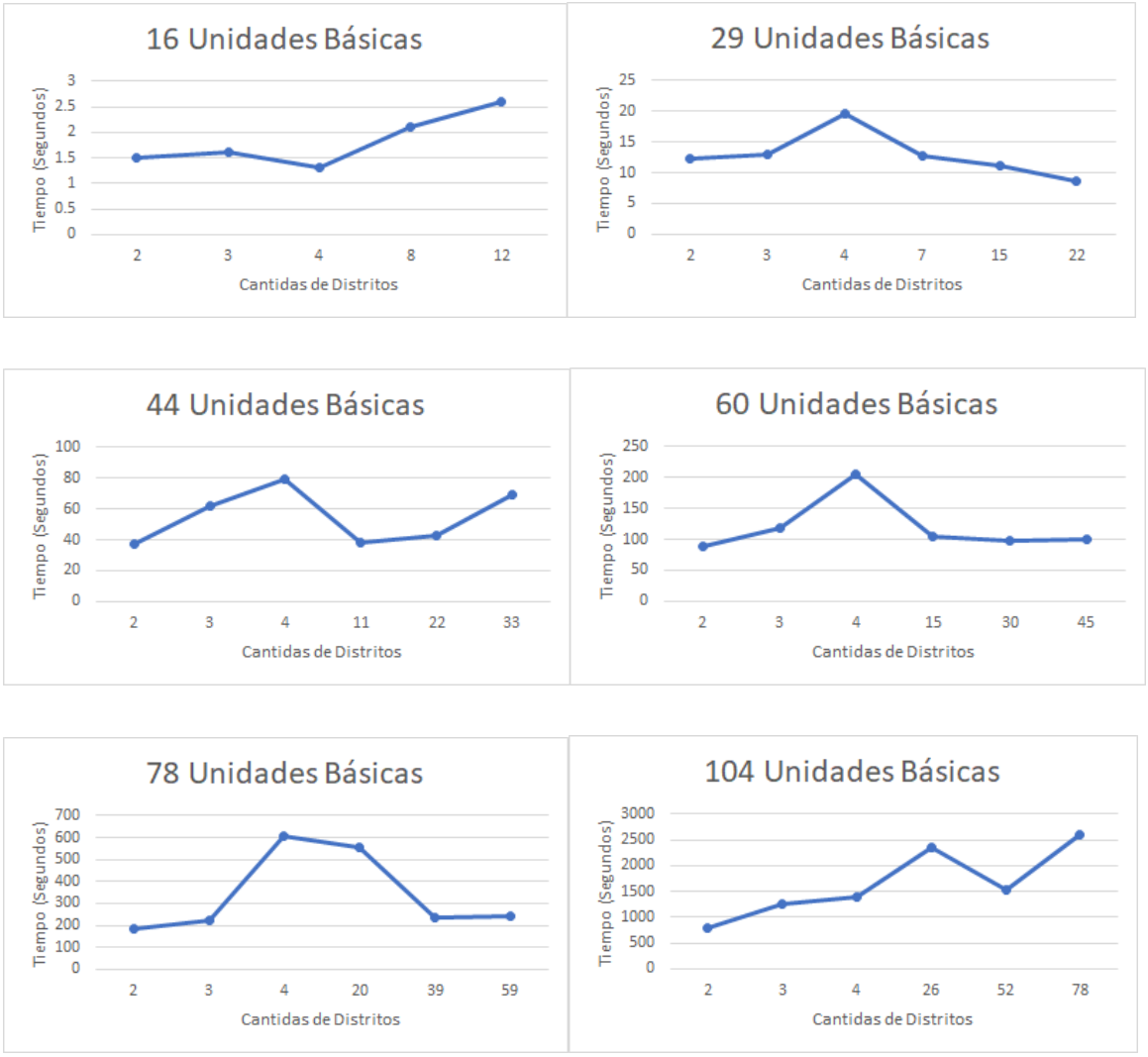

Figura 7. Gráficas comparativas del tiempo de ejecución (segundos) para 100 iteraciones.

Al observar cada gráfica del tiempo de procesamiento (**Figura 7**) se puede concluir que cada instancia tiene al menos una cantidad de unidades básicas críticas, en la cual se incrementa el tiempo de procesamiento; adicionalmente, en términos generales mientras más unidades básicas tiene la instancia, mayor será el tiempo de procesamiento promedio (Ver **Tabla 4**).

Tabla 4.
Tiempo de procesamiento Promedio de cada instancia con la Versión 2 (T=100 iteraciones).

Instancia	Tiempo Promedio (segundos)
16	1.82
29	12.9
44	54.8
60	119.3
78	342.9
104	1657.7
107	850.4

Se ejecutó un experimento con cantidad de iteraciones más elevadas (T=1000, T=2000 y T=5000). Los resultados se muestran en la **Tabla 5**. En dicha tabla se puede observar que para las instancias de más de 60 UBs, el heurístico tiene un tiempo de procesamiento elevado, y por tanto no fue posible tomar datos de resultados computacionales. Para la instancia de 78 UBs solo fue posible hacerlo para un número de distritos pequeño. Y para las instancias de 104 y 270 UBs el tiempo de cómputo es prohibitivo por lo tanto no se muestran en la tabla.

Tabla 5.
Resultados de los experimentos computacionales para la versión 2 (T=1000, 2000 y 5000 Iteraciones)

		ITERACIONES					
		1000		2000		5000	
Cantidad BU	**Distritos**	**Distancia (Km):**	**Tiempo Ejecución (s)**	**Distancia (Km):**	**Tiempo Ejecución (s):**	**Distancia (Km):**	**Tiempo Ejecución (s):**
16	2	223.14	22.2	223.14	37.6	223.14	106.4
	3	112.20	25.5	112.20	29.2	112.20	106.2
	4	67.95	32.1	67.95	26.7	67.95	89.8
	8	16.95	23.8	16.95	39.2	16.95	72.1
	12	6.06	29.9	6.06	35.1	6.06	70.2
29	2	240.63	90.1	227.39	163.6	233.76	519.5
	3	138.32	96.8	136.08	582.7	138.73	779.6
	4	85.67	159.5	82.75	223.8	73.43	528.1
	7	23.25	132.2	23.30	204.1	22.33	527.4
	15	4.37	154.2	4.37	228.4	4.37	567.4
	22	1.58	101.9	1.58	219.0	1.58	529.1
44	2	806.45	418.2	782.12	451.6	779.35	1 209.0
	3	513.45	3 862.5	495.46	4 901.0	494.82	8 630.8
	4	351.76	10 122.9				
	11	47.39	270.7				

	22	10.13	326.5
	33	3.63	319.5
60	2	1 481.43	817.9
	3	912.73	1 172.0
	4	689.98	55 776.6
	15	54.09	931.3
	30	10.31	912.2
	45	3.22	893.1
78	2	3 386.77	2 603.0
	3	2 194.10	3 484.0

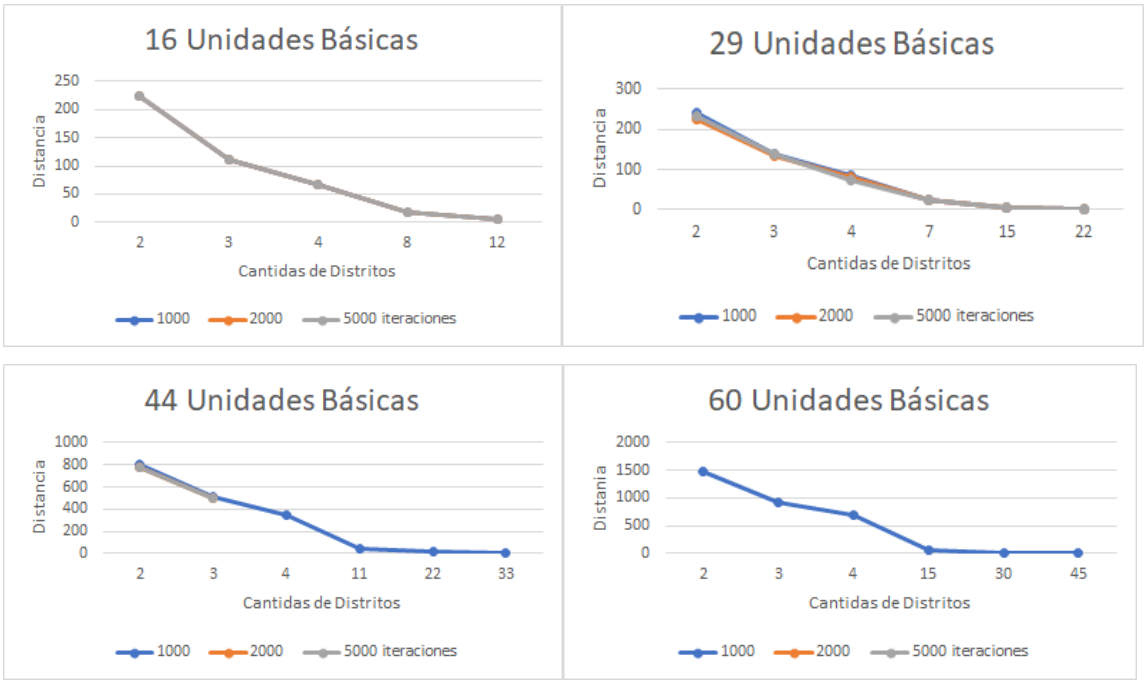

Figura 8. Gráficas comparativas de la distancia entre unidades básicas variando la cantidad de iteraciones entre 1000, 2000 y 5000

Como se puede observar en la **Figura 8**, al comparar los resultados con una cantidad diferente de iteraciones (1000, 2000, 5000) no hay diferencia significativa en la distancia total entre unidades básicas.

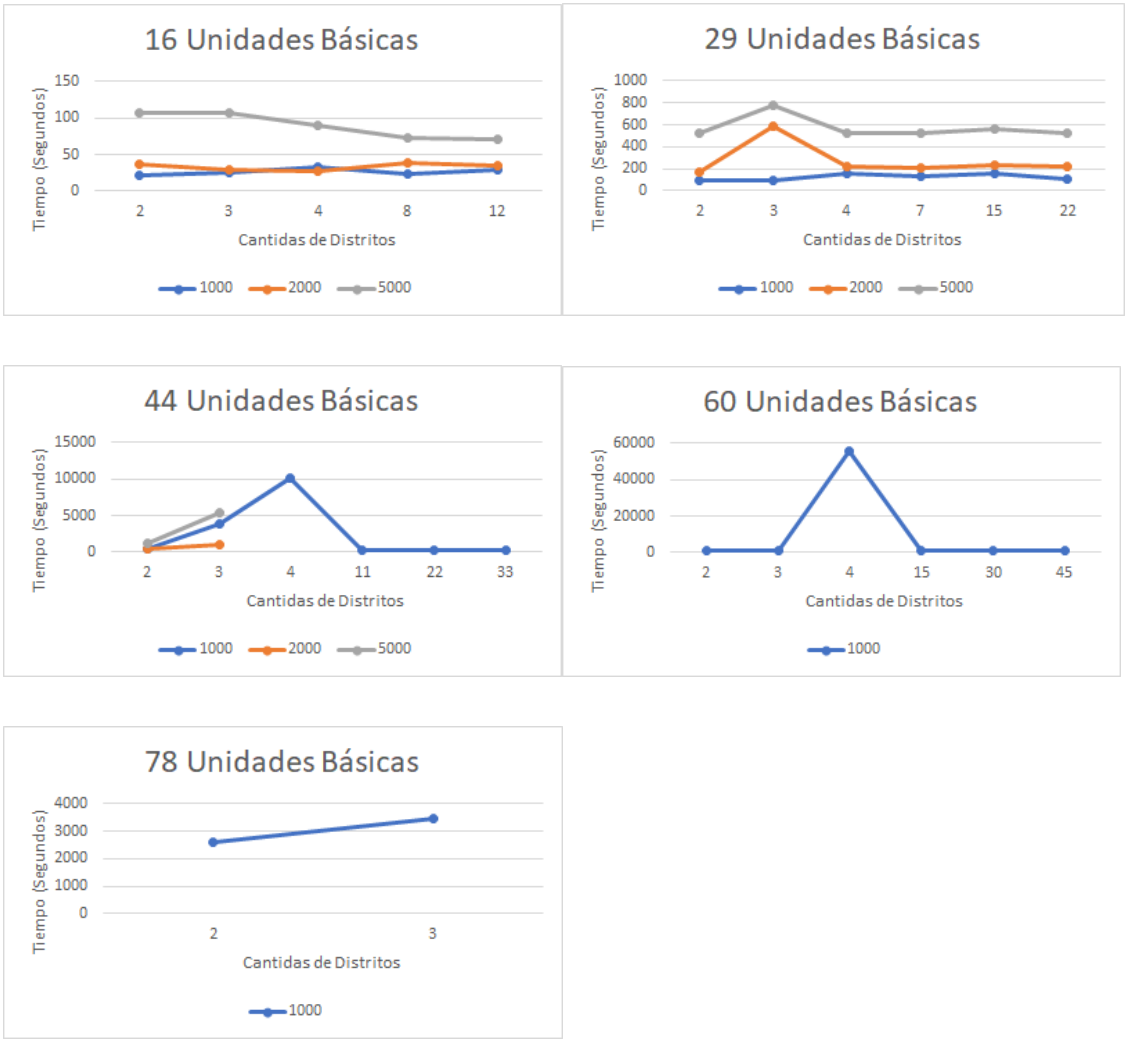

Figura 9. Gráficas comparativas del tiempo de ejecución (segundos) variando la cantidad de iteraciones entre 1000, 2000 y 5000

Se puede observar que existen diferencias en el tiempo de ejecución variando la cantidad de unidades básicas. También al comparar los diferentes gráficos, se puede observar que a medida

que crece la cantidad de iteraciones se incrementa el tiempo de ejecución, llegando incluso a ser exagerado o prohibitivo. La **Figura 10** resume el comportamiento de la distancia para cada instancia teniendo en cuenta todos los experimentos realizados anteriormente.

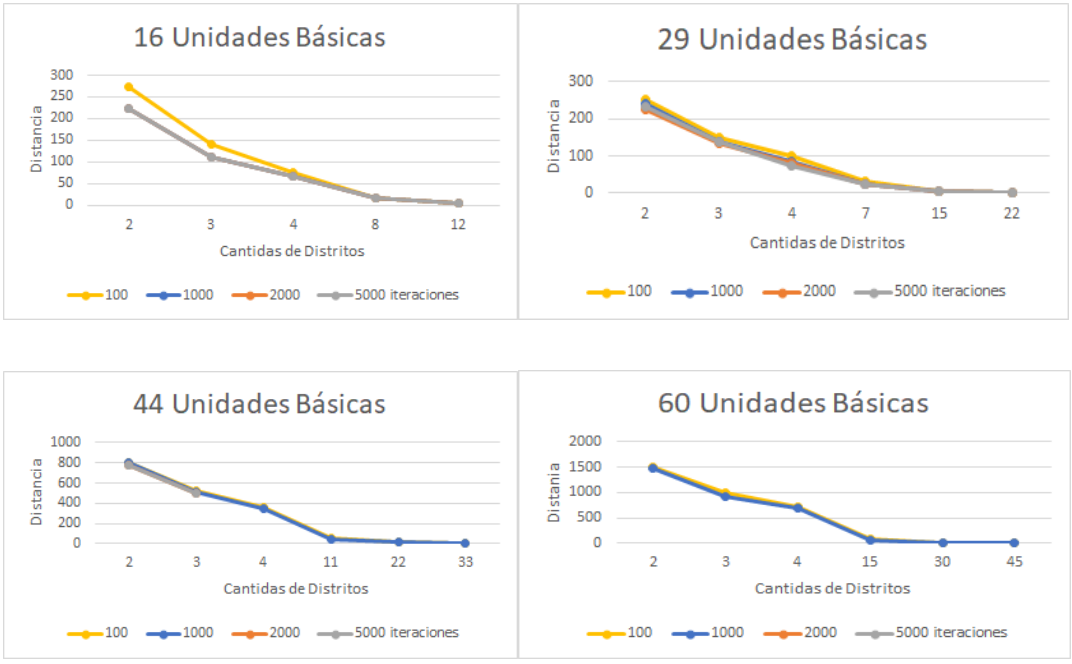

Figura 10. Gráficas comparativas de la distancia entre unidades básicas variando la cantidad de iteraciones entre (100, 1000, 2000 y 5000)

Al analizar la **Figura 10**, se observa que existe una diferencia entre la distancia con T=100 iteraciones en las instancias de 16 y 29 unidades básicas. Con las iteraciones grandes (T=1000, 2000, y 5000) se logran menores distancias. En el resto de los gráficos se observa que se logran resultados similares con pocas y con muchas iteraciones.

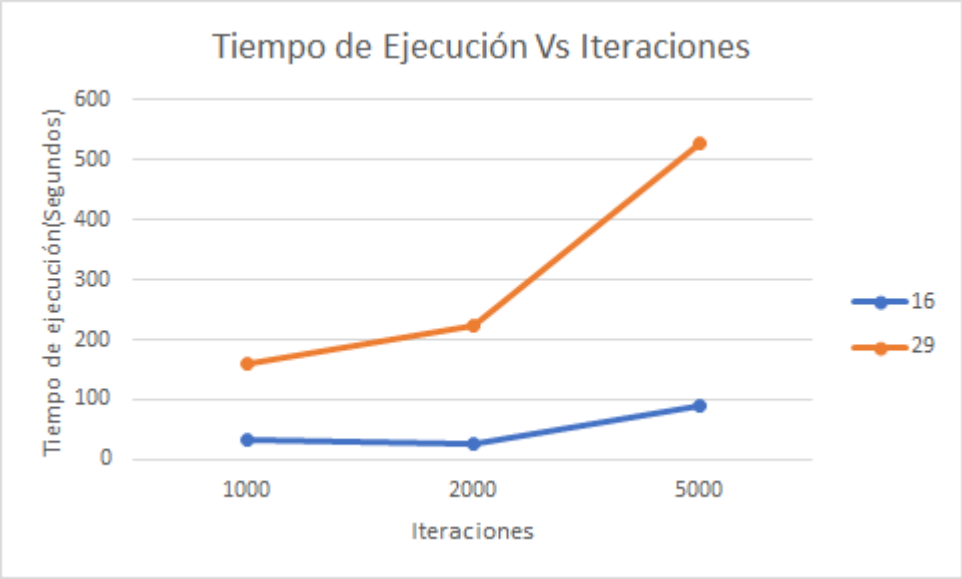

Figura 11. Gráficas comparativas del tiempo de ejecución (segundos) variando la cantidad de iteraciones para 4 distritos

Tal como se puede observar en la **Figura 11**, el tiempo de ejecución se incrementa a medida que se tiene mayor cantidad de iteraciones y también mayor cantidad de unidades básicas, lo cual es consistente en la medida en que se requiere un esfuerzo computacional mayor.

Finalmente, para analizar el efecto del preprocesador del optimizador se analizó la depuración de distritos redundantes y dominados que este realiza antes de iniciar con el proceso de optimización. En la **Tabla 6** se puede ver la cantidad de distritos borrados para cada instancia cuando se corrió con 1.000 iteraciones.

Tabla 6.
Depuración de distritos del preprocesador Gurobi para 4 distritos

	1.000 Iteraciones	
Cantidad BU	**Distritos Originales**	**Distritos Borrados**
16	79 897	52 593
29	259 778	77 532
44	549 738	97 257
60	990 761	118 686
78	1 624 856	134 114

7.3 Comparación con otros trabajos de la literatura

Uno de los objetivos planteados inicialmente es la comparación de MSH con los resultados de otras investigaciones. Para ello se utilizaron las mismas instancias utilizadas por Cortes (2018) en un optimizador comercial (Xpress), los resultados de ambos se presentan en la **Tabla 7**. Es preciso aclarar que los experimentos con el optimizador comercial se ejecutaron durante un tiempo máximo de una hora, lo cual explica por qué en muchos de los experimentos con instancias grandes no se obtuvo un resultado con Xpress.

Tabla 7.
Comparación de resultados MSH con los resultados de Optimización Xpress

			Solución con Xpress		Solución MSH	
Referencia	UB	Distritos	Distancia (Km)	Tipo de Solución	Distancia MSH (Km)	Diferencia Porcentual
Comunas	16	2	223.14	Óptima	223.14	0%
		3	112.18	Óptima	112.20	0%
		4	67.95	Óptima	67.95	0%
		8	20.14	Entera	16.95	-16%
		12	6.86	Entera	6.06	-12%
Copacabana	29	2	206.39	Óptima	227.39	10%
		3	103.92	Óptima	136.08	31%
		4	73.95	Entera	73.43	-1%
		7	54.42	Entera	22.33	-59%
		15	7.99	Entera	4.37	-45%
		22	1.61	Entera	1.58	-2%
Envigado	44	2	651.82	Entera	779.35	20%
		3	454.36	Entera	494.82	9%
		4	-	Sin Solución	351.76	
		11	-	Sin Solución	47.39	
		22	24.78	Entera	10.13	-59%
		33	5.14	Entera	3.63	-29%
Itagüí	60	2	1 130.11	Entera	1 481.43	31%
		3	-	Sin Solución	912.73	
		4	-	Sin Solución	689.98	
		15	-	Sin Solución	54.09	
		30	-	Sin Solución	10.31	

		45	-	Sin Solución	3.22	
Bello	78	2	6 372.36	Entera	3 383.65	-47%
		3	-	Sin Solución	2 194.10	
		4	-	Sin Solución	1 623.72	
		20	-	Sin Solución	125.75	
		39	-	Sin Solución	15.10	
		59	-	Sin Solución	4.21	
Envigado-Itagüí	104	2	12 590.70	Entera	7 492.23	-40%
		3	-	Sin Solución	4 810.49	
		4	-	Sin Solución	3 581.01	
		26	-	Sin Solución	188.48	
		52	-	Sin Solución	22.90	
		78	-	Sin Solución	6.49	
Bello-Copacabana	107	2	-	Sin Solución		
		3	-	Sin Solución		
		4	-	Sin Solución		
		27	-	Sin Solución		
		54	-	Sin Solución		
		80	-	Sin Solución		
Medellín	270	2	-	Sin Solución		
		3	-	Sin Solución		
		4	-	Sin Solución		
		68	-	Sin Solución		
		135	-	Sin Solución		
		203	-	LP No optimizado		

Al analizar las cifras de la **Tabla 7** comparando los resultados obtenidos con MSH versus los resultados del optimizador comercial Xpress, se observa lo siguiente:

- Cuando Xpress obtiene una solución óptima, MSH en algunos casos logra encontrar la misma solución y en otros entrega una solución aproximada. En los resultados obtenidos, la solución MSH más alejada de la óptima estuvo un 31% por encima.
- Para las instancias de 16 y 29 BU para las cuales se tienen todos los datos con ambas soluciones, se puede apreciar que en los casos en los cuales el optimizador comercial obtuvo una solución entera, se logró una solución similar o incluso mejor con MSH.
- A medida que se incrementa el tamaño de la instancia es más difícil obtener una solución con el optimizador comercial porque la complejidad del problema sobrepasa la capacidad de cómputo del equipo; pero con MSH (ejecutándolo con un número reducido de iteraciones) es posible obtener una solución en todos los casos.

8 Conclusiones y trabajos Futuros

Los problemas de distritación agrupan una gran cantidad de necesidades de la industria, del sector servicios y de las entidades gubernamentales para tomar decisiones claves en sus negocios. A lo largo de la historia, muchos autores han formulado diversos métodos para dar soluciones particulares a los problemas de distritación relacionados por ejemplo con la asignación de clientes a personal de ventas, agrupamiento de pacientes para la atención en la industria de atención hospitalaria en casa, ubicación de centros de distribución, agrupamiento de entidades administrativas para la asignación de recursos gubernamentales, etc.

En este trabajo se realizó exitosamente la implementación del metaheurístico MSH y se demostró que es posible su utilización para la solución de problemas de distritación, luego de experimentar con problemas de diferentes tamaños. Los resultados se pueden aplicar a cualquier problema de distritación.

Al incrementar la cantidad de iteraciones durante la solución propuesta con MSH, se mejora levemente la calidad de la solución. Pero, una cantidad muy grande de iteraciones (mayor de 2000) tiende a incrementar de manera excesiva el tiempo de ejecución de la solución.

Es necesario tener en cuenta que para utilizar el metaheurístico multi-space sampling (MSH) como solución a problemas de distritación, es factible encontrar soluciones con un número de iteraciones grande (mayor a 1000) cuando se trabaja con instancias inferiores o iguales a 78 unidades básicas. Para problemas de mayor tamaño se podría utilizar MSH con pocas iteraciones (alrededor de 100).

Al comparar los resultados de MSH con los valores óptimos en cada una de las instancias de prueba, se puede concluir que el MSH proporciona una solución cercana a la óptima; e incluso, cuando el optimizador comercial sólo es capaz de encontrar una solución entera, el MSH logra una mejor solución; esto significa que MSH puede llegar a ser un excelente complemento de un optimizador comercial para solucionar problemas de la industria.

Si se dispone de un optimizador comercial, el MSH podría ser implementado como un método alterno, ya que proporciona soluciones incluso en aquellos escenarios en los cuales no es posible encontrar una solución factible con el optimizador comercial, o en los casos en los cuales el tiempo de ejecución del optimizador comercial es demasiado elevado. En los casos en los cuales no se disponga de un optimizador comercial, MSH podría ser una excelente alternativa de solución.

Para futuras investigaciones sería conveniente desarrollar un solver (u optimizador) para integrarlo al algoritmo MSH en la fase de ensamble y de esta forma, no depender de ningún software comercial. Esto permitiría realizar implementaciones, y dar solución a las necesidades de la industria con una herramienta construida totalmente al interior de la universidad. Otro trabajo interesante sería realizar implementaciones con el metaheurístico MSH desarrollado, con el fin de optimizar un proceso del sector real.

Referencias

Bação, F., Lobo, V., & Painho, M. (2005, May). Self-organizing maps as substitutes for k-means clustering. In *International Conference on Computational Science* (pp. 476-483). Springer, Berlin, Heidelberg.

Benzarti, E., Sahin, E., & Dallery, Y. (2013). Operations management applied to home care services: Analysis of the districting problem. *Decision Support Systems*, *55*(2), 587-598.

Bergey, P. K., Ragsdale, C. T., & Hoskote, M. (2003). A decision support system for the electrical power districting problem. *Decision Support Systems*, *36*(1), 1-17.

Bergey, P. K., Ragsdale, C. T., & Hoskote, M. (2003). A simulated annealing genetic algorithm for the electrical power districting problem. *Annals of Operations Research*, *121*(1), 33-55.

Blais, M., Lapierre, S. D., & Laporte, G. (2003). Solving a home-care districting problem in an urban setting. *Journal of the operational research society*, *54*(11), 1141-1147.

Bozkaya, B., Erkut, E., & Laporte, G. (2003). A tabu search heuristic and adaptive memory procedure for political districting. *European journal of operational research*, *144*(1), 12-26.

Cabrera, N., Cordeau, J. F., & Mendoza, J. E. (2022). The doubly open park-and-loop routing problem. *Computers & Operations Research*, 143, 105761.

Caro, F., Shirabe, T., Guignard, M., & Weintraub, A. (2004). School redistricting: embedding GIS tools with integer programming, *Journal of the Operational Research Society* 55 (2004) 836–849.

Cortés, Sebastián. (2018). Modeling and solution of districting problems in home healthcare. Tesis de Maestría en Ingeniería. Universidad de Antioquia.

Cortés, S., Gutiérrez, E. V., Palacio, J. D., & Villegas, J. G. (2018, October). Districting Decisions in Home Health Care Services: Modeling and Case Study. In *Workshop on Engineering Applications* (pp. 73-84). Springer.

D'Amico, S. J., Wang, S. J., Batta, R., & Rump, C. M. (2002). A simulated annealing approach to police district design. *Computers & operations research*, *29*(6), 667-684.

Drexl A, Haase K (1999) Fast approximation methods for sales force deployment. *Manag Sci* 45:1307–1323

Duque, J. C., Church, R. L., & Middleton, R. S. (2011). The p-Regions Problem. p-区域问题. *Geographical Analysis*, 43(1), 104-126.

Easingwood, C. (1973). A heuristic approach to selecting sales regions and territories, *Journal of the Operational Research Society*, *24*(4), 527-534.

Ferland, J. A., & Guénette, G. (1990). Decision support system for the school districting problem. *Operations Research*, *38*(1), 15-21.

Fleischmann, B., & Paraschis, J. N. (1988). Solving a large scale districting problem: a case report, *Computers & Operations Research*, *15*(6), 521-533.

Forman, S. L., & Yue, Y. (2003, July). Congressional districting using a TSP-based genetic algorithm. In *Genetic and Evolutionary Computation Conference* (pp. 2072-2083). Springer, Berlin, Heidelberg.

Gómez, A., Mariño, R., Akhavan-Tabatabaei, R., Medaglia, A. L., & Mendoza, J. E. (2016). On modeling stochastic travel and service times in vehicle routing. *Transportation Science*, 50(2), 627-641

Hess, S. W., & Samuels, S. A. (1971). Experiences with a sales districting model: criteria and implementation. *Management Science*, *18*(4-part-ii), P-41.

Kalcsics, J., Nickel, S., Puerto, J., & Rodríguez-Chía, A. M. (2015). Several 2-facility location problems on networks with equity objectives. *Networks*, *65*(1), 1-9.

Lahrichi, N., Lapierre, S. D., Hertz, A., Talib, A., & Bouvier, L. (2006). Analysis of a territorial approach to the delivery of nursing home care services based on historical data. *Journal of medical systems*, *30*(4), 283-291.

Lodish, L. M. (1975). Sales territory alignment to maximize profit. *Journal of Marketing Research*, *12*(1), 30-36.

Mendoza, J. E., & Villegas, J. G. (2013). A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, *7*(7), 1503-1516.

Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2016). A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, *70*, 113-128.

Osman, I. H., & Kelly, J. P. (1997). Meta-heuristics theory and applications. *Journal of the Operational Research Society*, *48*(6), 657-657

Ricca, F., & Simeone, B. (2008). Local search algorithms for political districting. *European Journal of Operational Research*, *189*(3), 1409-1426.

Ríos-Mercado, R. Z., & Bard, J. F. (2019). An exact algorithm for designing optimal districts in the collection of waste electric and electronic equipment through an improved reformulation. *European Journal of Operational Research*, *276*(1), 259-271.

Ríos-Mercado, R. Z., & Escalante, H. J. (2016). GRASP with path relinking for commercial districting. *Expert Systems with Applications*, *44*, 102-113.

Ríos-Mercado, R. Z., & Fernández, E. (2009). A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research*, *36*(3), 755-776.

Salazar-Aguilar, M. A., Ríos-Mercado, R. Z., & González-Velarde, J. L. (2013). GRASP strategies for a bi-objective commercial territory design problem. *Journal of Heuristics*, *19*(2), 179-200.

Skiera B, Albers S (1994) COSTA: Ein Entscheidungs-Unterstützungs-System zur deckungsbeitragsmaximalen Einteilung von Verkaufsgebieten. Z Betriebswirt 64:1261–1283

Zoltners, A. A., & Sinha, P. (2005). The 2004 ISMS Practice Prize Winner—Sales territory design: Thirty years of modeling and implementation. *Marketing Science*, *24*(3), 313-331.