Localización Óptima de Colegios en Medellín: Un Enfoque de Optimización Lineal

Camilo Bermúdez Colorado, Jesús David Cantellón Espitia, Luis Alejandro Baena Marin

Universidad EAFIT. Escuela de Ciencias Aplicadas e Ingenierías.

Resumen

Medellín, como ciudad en constante crecimiento, enfrenta el reto de brindar una educación de calidad a todos sus habitantes. La ubicación estratégica de los colegios juega un papel fundamental en el acceso a la educación, especialmente para las poblaciones más vulnerables. Este trabajo propone la aplicación del lenguaje de programación `Python` para resolver el problema de Localización p-mediano capacitado, que busca determinar la ubicación óptima de los colegios en Medellín. En nuestra implementación buscamos minimizar la distancia ponderada total entre los estudiantes y los colegios, considerando factores como la capacidad de los colegios, la calidad y la accesibilidad. Se plantea un problema de programación lineal entera, donde se usa el algoritmo *Simulated Annealing* (en español se le llama *recocido simulado* y se abrevia como *SA*). Se espera que la implementación de este modelo contribuya a mejorar la accesibilidad a la educación siendo un estimulo en la planeación urbana para la consciente optimalidad de construcción de establecimientos educativos, especialmente en zonas con baja oferta educativa y optimizar la inversión pública en infraestructura educativa. Al final del proyecto, se van a comparar los resultados obtenidos con la distribución actual de colegios en Medellín.

Palabras Claves: *Educación, Optimización, Planificación, Problema p-mediano, Python, Simulated Annealing*

Abstract: Medellín, as a city in constant growth, faces the challenge of providing quality education to all its inhabitants. The strategic location of schools plays a fundamental role in access to education, especially for the most vulnerable populations. This paper proposes the application of the Python programming language to solve the capacitated p-median Location problem, which seeks to determine the optimal location of schools in Medellin. In our implementation we seek to minimize the total weighted distance between students and schools, considering factors such as school capacity, quality and accessibility. An integer linear programming problem is posed, where the Simulated Annealing algorithm is used. It is expected that the implementation of this model will contribute to improving accessibility to education by providing a stimulus for urban planning towards the conscious optimality of building educational facilities, especially in areas with low educational supply, and optimizing public investment in educational infrastructure. At the end of the project, the results obtained will be compared with the current distribution of schools in Medellín.

Keywords: *Education, Optimization, Planning, p-median problem, Python, Simulated Annealing*

1. Introducción

En Medellín, Colombia, garantizar el acceso a una educación de calidad para todos sus habitantes es un desafío continuo. La ubicación estratégica de los colegios juega un papel vital en este aspecto, especialmente para las poblaciones más vulnerables que enfrentan barreras de acceso a la educación.

El informe de calidad de vida de Medellín del 2020 de Medellín Cómo Vamos ([1], 2020) presenta que la tasa de deserción en secundaria puede estar asociada, entre otros determinantes, a condiciones físicas como la infraestructura de los colegios, la distancia de las residencias de los estudiantes a los establecimientos educativos y la falta de continuidad en la oferta educativa; observe la distancia como factor de deserción en este informe. La nota estadística del Análisis de accesibilidad a centros educativos del DANE ([2], 2022) nos informa que la literatura ha destacado que la distancia entre el hogar y los centros educativos es un factor importante en los procesos de educación, estando relacionado con la inasistencia escolar y la deserción estudiantil. Se han realizado análisis sobre este tema, donde se encontró que los niños y niñas que viven en zonas rurales o más lejos de la escuela primaria más cercana tienen más probabilidades de no asistir o desertar. Además, se ha investigado cómo esta distancia afecta la elección del medio de transporte y el desarrollo cognitivo de los estudiantes durante el trayecto entre el hogar y la escuela. También se ha observado cómo esta distancia impacta la experiencia educativa, especialmente para poblaciones vulnerables, contribuyendo a la inequidad educativa.

Por lo anteriormente presentado, la implementación de este modelo de optimización lineal tiene el potencial de generar un impacto positivo en la calidad de la educación en Medellín proveyendo un estimulo en la planeación urbana de construcción de nuevas instituciones educativas en el sentido que permita conscientizar sobre las posibles deficiencias en la distribución actual de los colegios y permita tomar medidas al respecto en las futuras construcciones de estos. Se espera que, a través de una mejor distribución de los colegios, se logre reducir la deserción escolar, mejorar el rendimiento académico y fomentar la equidad en la ciudad.

El presente documento se estructura de la siguiente manera: la sección 2 realiza una revisión de la literatura sobre optimización de la localización de instalaciones; la sección 3 se desarrolla el planteamiento del problema; la sección 4 describe el algoritmo utilizado; la sección 5 habla de la metodología usada en el estudio, detallando la implementación del modelo, los datos utilizados, la presentación de los resultados obtenidos al aplicar el modelo a nuestro caso de estudio y la comparación de las distancias ponderadas totales obtenidas con las distancias ponderadas totales de la distribución actual de colegios en la ciudad. Por ultimo, se concluye el estudio con líneas de trabajo futuro e implicaciones.

2. Estado del arte

La literatura presenta diversos métodos y enfoques para abordar este problema, algunos de los cuales se presentan a continuación:

Gac Serei e Isabel Margarita [3] (2006) presentan un modelo de optimización lineal determinístico para la localización óptima de colegios. En su modelo buscan maximizar el beneficio social, considerando la asignación de estudiantes a colegios basada en su ubicación y características socioeconómicas. A diferencia de modelos anteriores que solo minimizan costos de viaje, este modelo utiliza una función de utilidad multi-atributo que incluye calidad del colegio, costos de escolaridad y costos de viaje. Toman en cuenta la heterogeneidad tanto en la demanda, clasificando a los estudiantes por ubicación y características socioeconómicas, como en la oferta, diferenciando colegios privados, subvencionados y municipales. El modelo lo aplicaron a una comuna de Santiago de Chile, analizando diferentes escenarios y demostrando la importancia de una calibración adecuada de parámetros para reflejar las preferencias y capacidades de pago, lo que afecta directamente el beneficio de los agentes del sistema educativo.

Teqi Dai, Cong Liao y Shaoya Zhao [4] (2019) se centran en la optimización de la igualdad espacial en la educación primaria a través de la asignación aleatoria de plazas escolares, con el objetivo de desligar la relación entre la riqueza y la oportunidad educativa. Se enfrentan a la disparidad espacial en las oportunidades educativas, exacerbada por el sistema de asignación de escuelas

basado en la proximidad en China, que aumenta la desigualdad. Utilizan un algoritmo heurístico, específicamente la *Optimización por Enjambre de Partículas (PSO)*, para resolver el problema de programación cuadrática entera planteado por el nuevo modelo de optimización de asignación espacial. El modelo que propusieron, denominado modelo basado en RES, busca maximizar la semejanza de las distribuciones de probabilidad en los nodos de demanda, promoviendo así la igualdad de oportunidades educativas dentro de las restricciones de distancia máxima de viaje a la escuela y capacidades escolares. El estudio realizado en Beijing demuestra la eficacia del modelo en mejorar la igualdad espacial.

Pudy Prima y Aniati Murni Arymurthy [5] (2019) muestra un enfoque innovador para optimizar la ubicación y asignación de escuelas en el sur de Jakarta utilizando *Firefly Algorithm* (*FA*), un método de optimización metaheurística. El estudio aborda la política de zonificación del Sistema de Admisión de Nuevos Estudiantes de Indonesia, que tiene como objetivo minimizar la distancia de viaje de los estudiantes y promover una distribución equitativa de la calidad de las escuelas. Los autores modelan el problema como un problema p-mediano y emplean una representación individual *firefly* bidimensional para resolverlo. Su aplicación de *FA* redujo significativamente la distancia de viaje de los estudiantes en un 57,58 % en comparación con la condición previa a la zonificación, demostrando la efectividad del algoritmo para resolver problemas de optimización complejos en la planificación urbana.

Cong Liao, Bronte Scheuer, Teqi Dai y Yuan Tian [6] (2020) discuten el problema de la desigualdad espacial en las oportunidades educativas, particularmente cómo las familias con mayor poder adquisitivo pueden tener la oportunidad de acceder a mejores escuelas comprando propiedades dentro de ciertas zonas escolares. Proponen un modelo de optimización espacial que incorpora un mecanismo aleatorio en el sistema de asignación escolar basado en la proximidad para reducir tanto la desigualdad como la oposición potencial. El modelo tiene como objetivo minimizar la disparidad espacial de oportunidades educativas y la tasa de oposición potencial al asignar estudiantes a las 3 escuelas más cercanas dentro de las restricciones de capacidad. Desarrollaron un algoritmo heurístico para resolver el modelo, y lo aplicaron a un estudio de caso en Beijing, demostrando una mejora significativa en la igualdad espacial de oportunidades educativas.

Yulong Chen [7] (2021) realizó un estudio de optimización para la ubicación estratégica de escuelas primarias en áreas rurales de China, con el objetivo lograr un desarrollo educativo equilibrado. Abordó el desafío de minimizar los costos totales de transporte para los estudiantes, los costos de construcción de nuevas escuelas y los costos de construcción/mejora de carreteras, considerando la incertidumbre del tiempo de viaje debido a las condiciones variables de la carretera. Propuso un modelo de programación mixta entera, y utilizó un algoritmo híbrido de *Simulated Annealing* para resolver el problema de optimización, demostrando su aplicación a través de un estudio de caso práctico. La investigación destaca la importancia de las condiciones de la red de tráfico, que logran reducir significativamente el tiempo de viaje de los estudiantes.

Notamos la gran diversidad de enfoques en la optimización de la ubicación de colegios. Desde modelos determinísticos que consideran múltiples atributos socioeconómicos hasta heurísticas como *PSO* y metaheurísticas como *Firefly Algorithm*, cada estudio ofrece una perspectiva única para abordar este desafío. Además, hay modelos que incorporan mecanismos aleatorios y algoritmos como *Simulated Annealing* para adaptarse a diferentes contextos, desde áreas urbanas hasta rurales. Esta variedad demuestra la versatilidad y la capacidad de adaptación de la investigación en este campo. Estos estudios nos ofrece inspiración para ideas de enfoques y consideraciones sociales para el planteamiento de nuestro problema además de ideas de resolución, como el algoritmo *Simulated Annealing*. Procederemos a mostrar nuestro planteamiento del problema.

3. Planteamiento del Problema

El problema se puede formular como:

Minimizar
$$\sum_{i \in I} \sum_{j \in J} a_i d_{ij} X_{ij} \quad (1)$$

Sujeto a
$$\sum_{j \in J} X_{ij} = 1 \quad \forall i \in I \quad (2)$$
$$\sum_{j \in J} Y_j = p \quad (3)$$
$$\sum_{i \in I} a_i X_{ij} \leq c_j Y_j \quad \forall j \in J \quad (4)$$
$$X_{ij} \leq Y_j \quad \forall i \in I \quad \forall j \in J \quad (5)$$
$$X_{ij} \in \{0,1\} \quad \forall i \in I \quad \forall j \in J \quad (6)$$
$$Y_j \in \{0,1\} \quad \forall j \in J \quad (7)$$

Donde

i = Indice de los nodos de demanda

j = Indice de los nodos de localización potenciales

d_{ij} = Distancia mas corta entre el nodo i y el nodo j

p = Numero de colegios a ser localizados

a_i = Demanda en el nodo i

c_j = Capacidad colegio j

X_{ij} = Porcentaje de demanda i suplido por el colegio j (continuo)

$$Y_j = \begin{cases} 1, \text{ si el nodo j es seleccionado para colegio} \\ 0, \text{ en otro caso} \end{cases}$$

Figura 1: Planteamiento del PPLM

El problema de optimización de la ubicación de colegios se modela como un Problema de Programación Lineal Entera Mixta debido a la naturaleza continua de la variable de decisión X_{ij}; a su vez corresponde a la categoría de un problema p-mediano capacitado que se relaciona con la búsqueda de vértices p-medianos en una red (grafo). El objetivo es localizar p instalaciones (en nuestro problema son los colegios) de manera que se minimice la suma de las distancias desde cada vértice de demanda (clusteres de residencias de los estudiantes) hasta su instalación más cercana. El problema de p-medianos capacitados considera capacidades para el servicio que será proporcionado por cada mediano (es decir, cuantos estudiantes puede albergar el colegio). El servicio total demandado por los vértices identificados por los conglomerados de p-medianos no puede exceder su capacidad de servicio [8].

La función objetivo (1) busca minimizar la distancia ponderada total recorrida por los estudiantes. La ponderación de la distancia se hará de la siguiente manera: Dado un nodo de demanda i^* y un colegio j^*, la distancia ponderada será igual a la distancia euclidiana entre i^* y j^* menos el producto entre el estrato de i^* y la categoría de calidad educativa de j^*. Esta ponderación se hizo con base a que las familias en residencias pertenecientes a barrios con estratos altos no le prestan mucha importancia a la distancia real de su casa al colegio sino a la calidad del mismo; mientras que por otro lado, las familias de estratos mas bajo tienden a tener como prioridad la cercanía de los colegios a sus residencias.

Nótese que nuestro problema considera asignaciones parciales de cada nodo residencial a un colegio (variable de decisión X_{ij}), permitiendo que, si un grupo de estudiantes viven en un nodo, no todos sean necesariamente asignados a un solo colegio sino que se puedan dividir para que la demanda de un nodo pueda ser satisfecha por varios colegios.

Veamos la importancia de cada restricción: (2) el 100 % de la demanda de cada nodo sea suplida, (3) se localicen exactamente p colegios, (4) la capacidad de los colegios no sea superada, (5) no hayan estudiantes asistiendo a un colegio que no se seleccionó, (6) la variable X_{ij} sea continua entre 0 y 1 y (7) que Y_j sea binaria. Esta modelación permite una representación precisa de las decisiones involucradas en la planificación de la red escolar.

4. Algoritmo - Simulated Annealing

Simulated Annealing (*SA*) es una técnica de optimización metaheurística introducida por S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi ([9], 1983) para resolver el Problema del Viajante (*TSP*, por sus siglas en inglés).

Basándonos en [10] (2024) y [11] (2000), el algoritmo de *SA* se basa en el proceso de recocido utilizado en metalurgia, donde un metal se calienta a una temperatura alta y luego se enfría gradualmente. A altas temperaturas, los átomos se mueven rápido, y cuando la temperatura se reduce, su energía cinética también disminuye. Al final del proceso de recocido, los átomos caen en un estado más ordenado, y el material es más dúctil y más fácil de trabajar.

De manera similar, en *SA*, un proceso de búsqueda comienza con un estado de alta energía (una solución inicial) y gradualmente baja la temperatura (un parámetro de control) hasta alcanzar un estado de poca energía (solución).

SA se ha aplicado con éxito a una amplia gama de problemas de optimización, como *TSP*, plegamiento de proteínas, particionamiento de gráficos y programación de trabajos en talleres. La principal ventaja de *SA* es su capacidad para escapar de mínimos locales y converger hacia un mínimo

global. *SA* también es relativamente fácil de implementar y no requiere conocimiento a priori del espacio de búsqueda.

El proceso de *Simulated Annealing* comienza con una solución inicial y luego mejora iterativamente la solución actual perturbándola aleatoriamente y aceptando la perturbación con cierta probabilidad. La probabilidad de aceptar una solución peor es inicialmente alta y disminuye gradualmente a medida que aumenta el número de iteraciones.

El algoritmo de *SA* es bastante simple y se puede implementar de manera directa como se describe a continuación:

1. **Definir el problema a optimizar**. Esto involucra definir la función de energía, es decir, la función a minimizar o maximizar. Una vez la función de energía es definida, necesitamos establecer el valor de temperatura inicial y la solución candidata inicial. Esta última puede generarse aleatoriamente o utilizando algún otro método heurístico. Luego calculamos la energía de la solución candidata inicial.

2. **Definir la función de perturbación**. Se define una función de perturbación para generar nuevas soluciones candidatas.

3. **Criterio de aceptación**. El criterio de aceptación determina si se acepta o se rechaza una nueva solución. La aceptación depende de la diferencia de energía entre la nueva solución y la solución actual, así como de la temperatura actual. El criterio de aceptación clásico de *SA* proviene de la mecánica estadística y se basa en la distribución de probabilidad de Boltzmann. Un sistema en equilibrio térmico a una temperatura T puede encontrarse en un estado con energía E con una probabilidad proporcional a $e^{-E/kT}$ donde k es la constante de Boltzmann. Por lo tanto, a bajas temperaturas, hay una pequeña probabilidad de que el sistema esté en un estado de alta energía. Esto juega un papel crucial en *SA* porque un aumento en la energía permite escapar de mínimos locales y encontrar el mínimo global. Basado en la distribución de Boltzmann, el siguiente algoritmo define el criterio para aceptar una variación de energía ΔE a una temperatura T:

Algorithm 1 Función de aceptación

Datos: T, ΔE - La temperatura y la variación de energía entre la nueva solución candidata y la actual.
Resultado: Valor booleano que indica si la nueva solución es aceptada o rechazada.
if $(\Delta E < 0)$ **then**
  return True;
else
  $r \leftarrow$ generar un valor aleatorio en el rango $[0, 1)$
  if $(r < e^{-\Delta E/T})$ **then**
    return True
  else
    return False
  end if
end if

Una solución candidata con menor energía siempre es aceptada. Por el contrario, una solución candidata con mayor energía es aceptada al azar con probabilidad $e^{-\Delta E/T}$ (para nuestro propósito, podemos establecer $k = 1$). Este último caso se puede implementar comparando la probabilidad con un valor aleatorio generado en el rango $[0, 1)$.

4. **Programa de temperatura**. El programa de temperatura determina cómo cambia la temperatura del sistema con el tiempo. Al principio, la temperatura es alta para que el algoritmo pueda explorar una amplia gama de soluciones, incluso si son peores que la solución actual. A medida que aumentan las iteraciones, la temperatura disminuye gradualmente, por lo que el algoritmo se vuelve más selectivo. Un programa de temperatura simple se puede obtener multiplicando la temperatura actual por un factor α, que es menor que 1.

5. **Ejecutar el algoritmo *SA***. Finalmente, ejecuta el algoritmo aplicando de manera iterativa la función de perturbación y el criterio de aceptación. El algoritmo termina cuando la temperatura ha bajado a un cierto nivel T_{min}, cuando la energía de la solución actual es menor que un umbral fijo E_{th} o cuando ha realizado un número $iter\_max$ de iteraciones. A continuación, un pseudocódigo del proceso:

Algorithm 2 Optimización - *SA*

$T \leftarrow T_{max}$
$\mathbf{x} \leftarrow$ Genera la solución candidata inicial
$E \leftarrow E(\mathbf{x})$ Calcula la energía de la solución inicial
while $(iter <= iter\_max)$ **do**
  $\mathbf{x}_{new} \leftarrow$ Generar una nueva solución candidata
  $E_{new} \leftarrow$ Calcula la energia del nuevo candidato $\mathbf{x}_{new}$
  $\Delta E \leftarrow E_{new} - E$
  if `Accept`$(\Delta E, T)$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}_{new}$
    $E \leftarrow E_{new}$
  end if
  $T \leftarrow \alpha T$ Enfría la temperatura
end while
return x

La siguiente sección explica el tratamiento de datos y la implementación del algoritmo a nuestro problema.

5. Metodología

5.1. Obtención de datos

Obtuvimos los datos de la geometría de las manzanas del Geoportal DANE. Para la información concerniente a su uso (si es residencial o no), el estrato y al barrio al que pertenece utilizamos Mapas Medellín (que también se uso para obtener la dirección de los colegios de Medellín) y para la población de los barrios usamos la fuente Alcaldía de Medellín. Todos estos datos de las manzanas se guardaron en una sola base datos llamada `data.geojson`. Con respecto a las instituciones educativas, los datos de los matriculados por institución los obtuvimos del Ministerio de Educación Nacional y la calidad de los mismos se obtuvo de DataIcfes. Estos datos de los colegios se guardaron en el dataset `definitivo.xlsx`.

5.2. Tratamiento de datos

El código completo de la implementación, incluyendo la formulación del problema y la ejecución del algoritmo de Simulated Annealing, está disponible en el siguiente repositorio de GitHub: https://github.com/camilobdez/A-p-median-problem.

Importamos librerías y funciones necesarias:

Librerías:

- `Geopandas, Pandas, Numpy, Pulp`

Funciones:

- `LpProblem`, `LpVariable`, `lpSum`, `LpMinimize` y `LpStatus` de `pulp`
- `Point` de shapely.geometry
- `KMeans` de `sklearn.cluster`

Utilizamos la función `gpd.read_file()` para cargar datos geoespaciales desde el archivo en formato GeoJSON `data.geojson` (información espacial de todas las manzanas de Medellín) en un DataFrame de GeoPandas llamado `df`. Este DataFrame contiene las columnas `area` (área de la manzana), `MANZANA` (identificador de la manzana), `NOMBRE` (nombre del barrio), `BARRIO` (número del barrio), `ESTRATO` (estrato del barrio), `AREA_TOTAL_BARRIO` (área total del barrio), `POBLACION` (población estudiantil del barrio), `POBLACION_MANZANA` (población estudiantil de la manzana) y `GEOMETRY` (polígono).

Calculamos el número de clusters (grupos) para el algoritmo K-means ([12], 2007). Dividimos la longitud del DataFrame `df` entre 10 para obtener un número aproximado de clusters (793 se obtuvieron). Esto es una forma para reducir la dimensionalidad del problema. Ahora, calculamos los centroides de los polígonos de las manzanas en el DataFrame `df`. Los centroides son puntos representativos de los polígonos. Una vez hecho esto, extrajimos las coordenadas (x, y) de los centroides de las manzanas y las almacenamos en una lista llamada `coordinates`.

Inicializamos un objeto KMeans con el número de clusters obtenido anteriormente, establecimos una semilla aleatoria de 42 usando el atributo `random_state` para garantizar reproducibilidad, ajustamos el modelo K-means a los datos de coordenadas de los centroides, obtuvimos las etiquetas de cluster asignadas a cada manzana (cada etiqueta indica a qué cluster pertenece la manzana) y agregamos las etiquetas de cluster al DataFrame original `df` en una nueva columna llamada `cluster_label`. Ahora cada fila del DataFrame tiene una etiqueta que indica a qué cluster pertenece esa manzana.

Lo siguiente que hicimos fue ordenar el DataFrame `df` según la columna `cluster_label`, lo que significa que las filas se ordenarán de acuerdo con el valor de la etiqueta de cluster (esto agrupa las manzanas por cluster) y reiniciar los índices del DataFrame después de ordenarlo. Establecimos el parámetro `drop=True` para evitar que se agregue una columna adicional con los índices anteriores como una columna en el DataFrame resultante.

Agrupamos el DataFrame df por la columna `cluster_label` y luego aplicamos una función lambda que calcula el centroide de cada grupo de geometrías (centroides de los clusters) utilizando la función `unary_union` que se utiliza para unir todas las geometrías en cada grupo y calcular el centroide de la unión. Después, utilizamos los centroides calculados para crear un nuevo GeoDataFrame llamado `df_clusters`. La columna `geometry` de este DataFrame contendrá los centroides de los clusters, que representan los puntos centrales de cada cluster.

Ahora calculamos la suma de la población de todas las manzanas dentro de cada cluster, agrupamos el DataFrame `df` por la columna `cluster_label`, calculamos el valor medio del estrato de todas las manzanas dentro de cada cluster, agregamos la columna `poblacion` al GeoDataFrame `df_clusters` y le asignamos los valores de la suma de población de cada cluster (usando la función *values* que convierte la serie de `población_cluster` en un array `numpy`), agregamos la columna `estrato` al GeoDataFrame `df_clusters` y le asignamos los valores de la mediana del estrato de cada cluster, al igual que antes, usando la función `values`.

Cargamos nuestros otros datos desde el archivo Excel llamado `definitivo.xlsx` en un DataFrame llamado `df_colegios` que contiene las columnas e información `CODIGO_DANE`, `COLE_INST_NOMBRE`, `SECTOR`, `COLE_CATEGORIA` y `TOTAL_MATRICULA`.

Para la generación de los puntos propuestos del problema p-mediano, repetimos cada valor de las columna `COLE_CATEGORIA`, `TOTAL_MATRICULA` y poblacion diez veces y los convertimos en listas con nombres `calidad`, `capacidad` y `demanda`, respectivamente. Esto se hace porque para la propuestas de puntos iniciales para el problema p-mediano, consideraremos cada colegio actual 10 veces.

Luego generamos una lista `latitudes` de 5720 (572 colegios 10 veces) latitudes aleatorias entre 6,1925 y 6,35 (latitud máxima y mínima de Medellín), generamos otra lista `longitudes` de también 5720 longitudes aleatorias entre $-75{,}6461$ y $-75{,}5314$ (longitudes extremas de Medellín) y creamos una lista de objetos `Point` de `Shapely` a partir de las coordenadas de latitud y longitud generadas.

Construimos un GeoDataFrame llamado `gdf_puntos` con las coordenadas de los puntos aleatorios y el mismo sistema de referencia de coordenadas (CRS) que el DataFrame original `df`. Después, inicializamos una lista vacía para almacenar las distancias entre los puntos aleatorios y los centroides de los clusters, recorrimos cada cluster en el DataFrame `df_clusters`: Para cada cluster, calculamos la distancia entre cada punto aleatorio y el centroide del cluster. Ajustamos la distancia multiplicandola por 1000 (para nivelar las ordenas de magnitud con el substraendo) y el estrato del cluster por la calidad del colegio correspondiente al punto (por lo descrito en la sección 3), añadimos la lista de distancias para cada cluster a la lista `m_distancias` y la convertimos en una matriz numpy `distancias`, que contendrá las distancias ajustadas para cada punto aleatorio con respecto a cada cluster.

Utilizamos `pulp.pulpTestAll()` para asegurarnos de que la instalación de `PuLP` funcione correctamente y establecimos `Gurobi` como el solver predeterminado para `PuLP`. `Gurobi` es un potente software de optimización que ofrece solvers eficientes para una amplia gama de problemas de optimización. Con esto, le indicamos a PuLP que use `Gurobi` para resolver el problema de optimización que formularemos más adelante.

5.3. Implementacion del algoritmo

Definimos la función `objective_function` que toma como entrada los índices seleccionados (puntos propuestos seleccionados), la matriz de distancias, la demanda de cada cluster y la capacidad de cada colegio. Creamos un objeto de problema de programación lineal (LP) llamado `prob` con el nombre `Optimal_Assignment` y el objetivo de minimizar. Definimos las variables de decisión para el problema de asignación. Estas variables indican

si se asigna o no la demanda `i` al punto de venta `j`. Definimos la función objetivo que minimiza la suma de las distancias ponderadas por las asignaciones. Con respecto a las restricciones, cada demanda debe ser asignada a exactamente un punto de venta y la suma de la demanda asignada a cada punto de venta no debe exceder su capacidad. Luego, se resuelve el problema de optimización y se retorna el valor de la función objetivo una vez que se ha resuelto el problema. Esto representará la distancia total de la asignación óptima.

Luego, definimos una función llamada `generate_initial_state` que genera una solución inicial para el problema de asignación, seleccionando aleatoriamente una ubicación de entre un grupo de 10 posibles para cada colegio. Esta función toma dos argumentos opcionales: el número total de escuelas (`num_schools`) y el número de posibles ubicaciones por escuela (`num_possible_locations_per_school`). Por defecto, estos valores se toman de las variables definidas anteriormente. La función inicializa una lista para almacenar las ubicaciones seleccionadas para cada escuela. Después, la función itera sobre el número total de escuelas y en cada iteración calcula una ubicación aleatoria dentro del rango correspondiente de 10 posibles ubicaciones para la escuela actual (`i`). Esto se logra multiplicando el índice de la escuela (`i`) por el número de posibles ubicaciones por escuela y luego sumando un valor aleatorio entre 0 y 9. La función retorna la lista de ubicaciones seleccionadas como la solución inicial. Cada ubicación representará el índice de la ubicación seleccionada para cada escuela.

Ahora, definimos otra función llamada `generate_neighbor` (función de peturbación) que genera un vecino del estado actual cambiando la ubicación de un colegio seleccionado por otro dentro del rango correspondiente de ubicaciones posibles. Esta función toma dos argumentos: el estado actual (una lista de índices de ubicaciones seleccionadas para cada colegio) y el número de posibles ubicaciones por colegio (en nuestro caso es 10). La función crea una copia del estado actual para modificarla sin afectar al original, elige aleatoriamente un índice de la lista de ubicaciones seleccionadas para perturbar, calcula el inicio del grupo de posibles ubicaciones al que pertenece la ubicación actual del colegio seleccionado, genera un nuevo índice dentro del rango correspondiente de posibles ubicaciones para el colegio seleccionado y se asegura de que el nuevo índice generado no sea igual al índice actual del colegio seleccionado. Por último, asigna el nuevo índice al colegio seleccionado en la nueva configuración y retorna la nueva configuración, que es un vecino del estado actual.

Establecemos `acceptance_probability` (criterio de aceptación) que es una función que calcula la probabilidad de aceptación de un nuevo estado en el algoritmo de *Simulated Annealing* y toma los argumentos: el costo del estado actual (`old_cost`), el costo del nuevo estado (`new_cost`) y la temperatura actual (`temperature`). Esta función funciona con la lógica descrita en 4: Si el nuevo costo es menor que el costo del estado actual, se acepta el nuevo estado con una probabilidad de 1 (certeza), por el contrario, si el nuevo costo es mayor o igual al costo del estado actual, se calcula la probabilidad de aceptación utilizando la fórmula del *Simulated Annealing* ya anteriormente vista.

Ahora establecemos `simulated_annealing` que es una función que toma como entrada la matriz de distancias, la demanda, la capacidad, y otros parámetros opcionales como el número máximo de iteraciones, la temperatura inicial y la tasa de enfriamiento. La función genera un estado inicial aleatorio usando `generate_initial_state` y calcula su costo usando `objective_function`. Ahora, inicializamos la mejor solución encontrada con el estado actual e inicializamos el mejor costo encontrado con el costo del estado actual. Iteramos sobre el número máximo de iteraciones y en cada iteración: calculamos la temperatura para esta iteración basada en la temperatura inicial (1,0) y la tasa de enfriamiento (que establecimos en 0,95 lo que reduce la temperatura en una tasa de 0,95 elevada al número de la iteración, este es el programa de temperatura), generamos un nuevo estado vecino perturbando el estado actual usando `generate_neighbor` y calculamos su costo. Si el nuevo costo es menor que el costo del estado actual, o si se cumple la probabilidad de aceptación dada por `acceptance_probability`, se acepta el nuevo estado. Actualizamos el estado actual al nuevo estado y también el costo. Si el nuevo costo es menor que el mejor costo encontrado hasta ahora, actualiza la mejor solución encontrada. Finalmente, retorna la mejor solución encontrada y su costo.

Por ultimo, ejecutamos el algoritmo *Simulated Annealing* llamando a la función `simulated_annealing` con la matriz de distancias `distancias`, la demanda `demanda` y la capacidad `capacidad`. Recordemos que esta función devuelve la mejor solución encontrada (`best_state`) y su costo (`best_cost`).

Nótese que en la declaración de `objective_function`, dado la metodología que se acaba de escribir, se definió con `pulp` la siguiente variante del problema establecido en la sección 3:

Minimizar $\sum_{i \in I} \sum_{j \in J} a_i d_{ij} X_{ij}$

Sujeto a

$$\sum_{j \in J} X_{ij} = 1 \qquad \forall i \in I$$

$$\sum_{i \in I} a_i X_{ij} \leq c_j \qquad \forall j \in J$$

$$X_{ij} \in \{0,1\} \qquad \forall i \in I \quad \forall j \in J$$

Donde

i = Indice de los nodos de demanda
j = Indice de los nodos de localización seleccionados
d_{ij} = Distancia mas corta entre el nodo i y el nodo j
a_i = Demanda en el nodo i
c_j = Capacidad colegio j
X_{ij} = Porcentaje de demanda i suplido por el colegio j (continuo)

Figura 2: Problema alternativo post simulated annealing

En donde solo consideramos las restricciones relacionadas a: el dominio de X_{ij} entre 0 y 1, que se asigne toda la demanda de cada punto y que la demanda asignada a un colegio sea menor o igual que su capacidad.

5.4. Ejecución y resultados

La ejecución se realizó tres veces. En la primera ejecución se seleccionó una solución inicial aleatoria (cuyo valor de la función objetivo fue de 3921,13) con un número máximo de iteraciones de 1000, temperatura inicial del 1,0 y tasa de enfriamiento inicial de 0,95. Se ejecutó y se obtuvo un resultado de $-1319{,}88$. Este estado se selecciona como solución inicial de una nueva iteración del algoritmo con el mismo número máximo de iteraciones, temperatura inicial y tasa de enfriamiento. El valor alcanzado por la siguiente iteración fue de $-2763{,}38$. Por ultimo, volvimos a tomar este estado como la solución inicial de otra nueva iteración con los mismos parámetros y se obtuvo en esta tercera un resultado de $-3199{,}76$. Este valor hallado es mejor que la función objetivo evaluada en el estado real de los colegios de Medellín que es de $-3121{,}55$. Cada una de las tres iteraciones se demoró entre 5 y 7 horas. A continuación se observa la evolución de la función objetivo en las 3000 iteraciones:

Figura 3: Comportamiento de la función objetivo en las 3 ejecuciones.

El estado encontrado se representa en el siguiente gráfico:

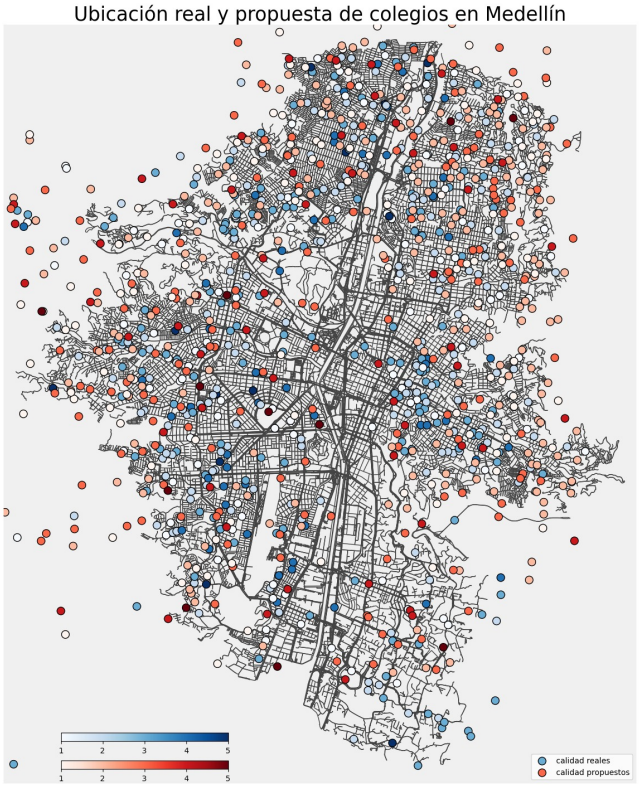

Figura 4: Mapa de Medellín con la posición de los colegios reales junto con su calidad (Azules) y con la posición de los colegios propuestos con su calidad (Rojos) donde las calidades 1, 2, 3, 4 y 5 son respectivamente D, C, B, A y A+.

Realizamos la ejecución 3 veces, siendo la solución inicial propuesta de cada una el mejor estado del anterior, y esto nos generó resultados mejores que los obtenidos en la distribución real en aproximadamente un 3 %. Aunque cabe aclarar que se puede seguir obteniendo resultados mejores si volvemos a hacer este proceso varias veces, aunque para objetivos de nuestro estudio, es suficiente hallar un estado que es mejor que el real de los colegios.

6. Conclusiones

Considerando los factores de calidad educativa de los colegios de Medellín, estrato de los punto de demanda y la distancia entre los estudiantes y los colegios, proveímos evidencia empírica para concluir que la distribución actual de los colegios de Medellín no es óptima habiendo distribuciones que retornan un mejor resultado en la función objetivo del problema considerado. Con nuestra implementación, encontramos una distribución que da una distancia ponderada 3 % menor que la distribución actual de los centros educativos. Como se mencionó en la introducción, el propósito de esta investigación era realizar una evaluación sobre la optimalidad en la localización de los colegios con el objetivo de realizar un primer paso al reconocimiento del problema o situación relacionada con la deficiente localización de las instituciones educativas. Este resultado puede servir como impulso a futuras investigaciones de planeación urbana sobre localización de colegios en la ciudad de Medellín al considerar que existen distribuciones mas eficientes y por lo tanto tomar en cuenta esto para los proyectos de construcción de colegios, siendo futuras lineas de investigación algoritmos que permitan determinar la localización óptima de nuevos colegios en planeación dado los actuales o una reubicación de los mismos. Una limitación de nuestro proyecto es que no toma en cuenta los medios de transporte, así que futuras investigaciones se podrían enfocar en considerar esta variable. Otra limitación es que debido al extenso tiempo de ejecución de cada iteración, no llegamos a un valor suficientemente cercano al óptimo, así que próximos estudios se podrían enfocar en este aspecto.

Referencias

[1] Medellín Cómo Vamos. *Educación Informe de Calidad de Vida de Medellín, 2020.* Accessed: 2024-05-20. 2020. URL: https://www.medellincomovamos.org/system/files/2021-09/docuprivados/Educaci%C3%B3n%20Informe%20de%20Calidad%20de%20Vida%20de%20Medell%C3%ADn%2C%202020.pdf.

[2] Departamento Administrativo Nacional de Estadística (DANE). *Análisis de Accesibilidad a Centros Educativos.* Accessed: 2024-05-20. 2022. URL: https://www.dane.gov.co/files/investigaciones/notas-estadisticas/abr_2022_nota_estadistica_analisis_accesibilidad_centros_educativos.pdf.

[3] Isabel Margarita Gac Serei. «Modelo de optimización lineal determinístico para la localización de colegios». Español. Tesis (magíster en gestión de operaciones)–Universidad de Chile, 2006. Tesis (ingeniero civil industrial)–Universidad de Chile, 2006. Bibliografía: h. 131-135. Santiago de Chile: Universidad de Chile. Facultad de Ciencias Físicas y Matemáticas. Escuela de Postgrados, 2006, págs. vi, 149.

[4] Teqi Dai, Cong Liao y Shaoya Zhao. «Optimizing the spatial assignment of schools through a random mechanism towards equal educational opportunity: A resemblance approach». En: *Computers, Environment and Urban Systems* 76 (jul. de 2019). Version of Record 23 March 2019, págs. 24-30. DOI: 10.1016/j.compenvurbsys.2019.03.004. URL: https://doi.org/10.1016/j.compenvurbsys.2019.03.004.

[5] Pudy Prima y Aniati Murni Arymurthy. «Optimization of school location-allocation using Firefly Algorithm». En: *Journal of Physics: Conference Series.* The 3rd International Conference on Computing and Applied Informatics 2018 18–19 September 2018, Medan, Sumatera Utara, Indonesia 1235 (2019). Published under licence by IOP Publishing Ltd, pág. 012002. DOI: 10.1088/1742-6596/1235/1/012002.

[6] Cong Liao et al. «Optimizing the spatial assignment of schools to reduce both inequality of educational opportunity and potential opposition rate through introducing random mechanism into proximity-based system». En: *Socio-Economic Planning Sciences* 72 (dic. de 2020). Version of Record 25

November 2020, pág. 100893. DOI: 10.1016/j.seps.2020.100893. URL: https://doi.org/10.1016/j.seps.2020.100893.

[7] Yulong Chen. «Optimizing Locations of Primary Schools in Rural Areas of China». En: *Volume 2021* (2021). Ed. por Fengtai Zhang. Open Access. DOI: 10.1155/2021/7573700.

[8] Luiz A.N. Lorena y Edson L.F. Senne. «A column generation approach to capacitated p-median problems». En: *Computers & Operations Research* 31.6 (mayo de 2004). FEG/UNESP, Universidade Estadual Paulista, Faculdade de Engenharia, Departamento de Matemática, Caixa Postal 205, 12515-410 Guaratinguetá, SP, Brazil, págs. 863-876. DOI: 10.1016/S0305-0548(03)00137-5. URL: https://www.sciencedirect.com/science/article/pii/S0305054803001375.

[9] S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi. «Optimization by Simulated Annealing». En: *Science* 220.4598 (mayo de 1983), págs. 671-680. DOI: 10.1126/science.220.4598.671. URL: https://doi.org/10.1126%2Fscience.220.4598.671.

[10] Baeldung. *Simulated Annealing Explained*. Ed. por Michal Aibin. Last updated: March 18, 2024. Baeldung. 2024. URL: https://www.baeldung.com/cs/simulated-annealing.

[11] Fernando Chiyoshi y Roberto D. Galvão. «A statistical analysis of simulated annealing applied to the p-median problem». En: *Annals of Operations Research* 96.1 (nov. de 2000), págs. 61-74. DOI: 10.1023/A:1018982914742.

[12] A.A. Chaves, F. de Assis Correa y L.A.N. Lorena. «Clustering Search Heuristic for the Capacitated p-Median Problem». En: *Innovations in Hybrid Intelligent Systems*. Ed. por E. Corchado, J.M. Corchado y A. Abraham. Vol. 44. Advances in Soft Computing. Berlin, Heidelberg: Springer, 2007. ISBN: 978-3-540-74971-4. DOI: 10.1007/978-3-540-74972-1_19. URL: https://doi.org/10.1007/978-3-540-74972-1_19.

[13] Isabel Gac, Francisco Martinez y Andres Weintraub. «Modelo De Optimización Lineal Determinístico Para La Localización De Colegios». En: *Revista Ingeniería de Sistemas* XX (2006), págs. 25-43. URL: https://dii.uchile.cl/~ris/RISXX/IGac_RIS.pdf.

[14] *Banco documentos - Alcaldía de Medellín*. Alcaldía de Medellín. URL: https://www.medellin.gov.co/es/centro-documental/proyecciones-poblacion-viviendas-y-hogares/.

[15] *Datos abiertos - Ministerio de Educación Nacional*. Ministerio de Educación Nacional. URL: https://www.mineducacion.gov.co/portal/micrositios-institucionales/Modelo-Integrado-de-Planeacion-y-Gestion/Datos-abiertos/349303:Datos-Abiertos.