UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Algoritmos evolutivos para el diseño de redes de saneamiento en Latinoamérica

Informe de Proyecto de Grado presentado por

Lucas Barbachan, Nicolás Herrera

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Sergio Nesmachnow
Renzo Massobrio

Montevideo, Setiembre de 2025

Barbachan, Lucas - Herrera, Nicolás

Proyecto de Grado / Lucas Barbachan, Nicolás Herrera. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2025.

VI, 89 p. 29, 7cm.

Directores:

Sergio Nesmachnow

Renzo Massobrio

Proyecto de Grado de Ingeniería – Universidad de la República, Programa en Ingeniería en Computación, 2025.

Referencias bibliográficas: p. 88 – 89.

1. Algoritmos Evolutivos, 2. Redes de saneamiento, 3. Optimización, 4. Infraestructura urbana, 5. Greedy. I. Nesmachnow, Sergio, Massobrio, Renzo, . II. Universidad de la República, Programa de Posgrado en Ingeniería en Computación. III. Título.

RESUMEN

En esta tesis se aborda el problema del diseño de redes de saneamiento en entornos urbanos mediante el uso de algoritmos evolutivos. Se trata de un desafío complejo de optimización, que involucra múltiples restricciones técnicas, topográficas y económicas. Se desarrolló un algoritmo evolutivo capaz de generar soluciones factibles y de bajo costo, utilizando datos geoespaciales y demográficos reales. El enfoque fue validado en tres escenarios representativos de Montevideo, donde demostró mejoras significativas frente a un algoritmo greedy, alcanzando reducciones superiores a un 10 % en el costo total de la red.

Además, el algoritmo mostró un comportamiento robusto y consistente en todas las instancias analizadas. La metodología propuesta es aplicable a otras ciudades de Latinoamérica que cuenten con datos adecuados, y abre la posibilidad de futuras extensiones hacia escenarios más complejos y automatizados.

Palabras claves:
Algoritmos Evolutivos, Redes de saneamiento, Optimización, Infraestructura urbana, Greedy.

Tabla de contenidos

Capítulo 1

Introducción

El saneamiento es un aspecto fundamental del desarrollo urbano y de la salud pública. Un sistema de saneamiento eficiente permite la recolección, transporte y tratamiento adecuado de las aguas residuales, minimizando los riesgos para la salud humana y el impacto ambiental. En regiones de Latinoamérica, donde existen zonas urbanas con deficiencias en infraestructura básica, el diseño adecuado de redes de saneamiento es una necesidad urgente para garantizar condiciones de vida dignas y sostenibles.

El diseño de redes de saneamiento implica una combinación de factores geográficos, hidrológicos, demográficos, normativos y económicos. Los ingenieros deben encontrar soluciones que respeten las restricciones físicas del terreno (como topografía y pendiente), cumplan con criterios técnicos (diámetros, velocidades de flujo, pendientes, profundidades) y sean viables desde el punto de vista económico. Esto genera un problema de optimización complejo y multidimensional.

Tradicionalmente, estos problemas se han abordado mediante métodos determinísticos, que si bien pueden brindar soluciones satisfactorias, presentan limitaciones cuando el espacio de búsqueda es amplio y no lineal. En este contexto, los algoritmos evolutivos (AE) surgen como una alternativa prometedora. Inspirados en la selección natural, los AE permiten explorar eficientemente grandes espacios de soluciones posibles, adaptándose a problemas con restricciones complejas y funciones objetivo no derivables ni continuas.

En esta tesis se explora el uso de algoritmos evolutivos para abordar el diseño de redes de saneamiento en escenarios urbanos de Latinoamérica. El trabajo se desarrolla sobre datos geoespaciales reales, procesados para cons-

truir escenarios representativos donde se aplican los algoritmos. Se propone una formulación del problema considerando aspectos topográficos, demográficos y normativos. A partir de esta formulación se desarrolla un modelo de optimización que permite evaluar la calidad de diferentes soluciones.

El principal objetivo de esta investigación es evaluar la capacidad de los AE para generar soluciones de alta calidad en términos de costo y factibilidad para el diseño de redes de saneamiento. Para ello, se construyó un algoritmo evolutivo específico y se comparó su desempeño con un enfoque greedy que simula una estrategia constructiva más clásica.

Los objetivos específicos que se plantean son:

- Formular el problema de diseño de redes de saneamiento como un problema de optimización con restricciones.
- Desarrollar y adaptar un algoritmo evolutivo para resolver el problema.
- Diseñar e implementar un sistema para la generación de escenarios realistas a partir de datos geoespaciales.
- Evaluar el desempeño del AE mediante experimentos sobre instancias reales.
- Comparar los resultados obtenidos con los de un algoritmo greedy.

La estructura del documento es la siguiente: en el Capítulo 2 se presenta el marco teórico necesario para comprender los fundamentos de los algoritmos evolutivos y del diseño de redes de saneamiento. En el Capítulo 3 se realiza un relevamiento del estado del arte, analizando trabajos previos y detectando oportunidades de mejora. El Capítulo 4 describe la metodología empleada para representar los escenarios y construir los algoritmos. En el Capítulo 5 se presenta el análisis experimental, que incluye la selección de parámetros, la evaluación del desempeño y la comparación con el enfoque greedy. Finalmente, el Capítulo 6 expone las consideraciones finales del trabajo.

Capítulo 2

Marco teórico

El diseño de redes de saneamiento es un desafío complejo que aborda dos subproblemas fundamentales:

1. Problema del diseño de redes: implica múltiples aspectos, incluyendo la dimensión y topografía del área, así como la disposición de los elementos que componen la red.
2. Optimización del diseño hidráulico: se enfoca en parámetros técnicos como el tamaño y pendiente de las tuberías, las profundidades de instalación y la eventual necesidad de estaciones de bombeo. Este subproblema está estrechamente vinculado al anterior.

Ambos subproblemas son complejos debido a la gran cantidad de variables involucradas. Factores como los costos de construcción, los materiales disponibles, las condiciones del terreno y la normativa vigente influyen directamente en las decisiones de diseño. El objetivo final es establecer un trazado y seleccionar componentes que garanticen un transporte eficiente de las aguas residuales, reduciendo al mínimo el riesgo de desbordes, fugas o derrames que puedan afectar el ambiente o la salud pública.

Además, es fundamental que la red esté dimensionada de forma adecuada para manejar los caudales esperados, incluyendo situaciones de mayor demanda o eventos extraordinarios. Esto requiere combinar criterios técnicos con una mirada integral del entorno urbano y social donde se implementará la solución.

En este capítulo se presentan los fundamentos teóricos que sustentan el desarrollo del trabajo. Se introducen los conceptos de algoritmos evolutivos y de redes de saneamiento, sus principales componentes y restricciones.

2.1. Fundamentos teóricos

En esta sección se presentan los fundamentos teóricos que se consideran relevantes para el desarrollo del proyecto, definiciones generales sobre las redes de saneamiento y algunas específicas para esta aplicación.

2.1.1. Algoritmos evolutivos

Los algoritmos evolutivos (AE) constituyen una familia de técnicas de optimización y búsqueda inspiradas en los principios de la selección natural y la evolución biológica. Estos algoritmos modelan una población de posibles soluciones, representadas como individuos, que evolucionan a lo largo de generaciones sucesivas mediante la aplicación de operadores genéticos como la selección, la recombinación y la mutación. A través de este proceso iterativo, los AE buscan encontrar soluciones óptimas o cercanas a óptimas, adaptándose de forma dinámica a las características del problema. En el libro Genetic Algorithms in Search, Optimization, and Machine Learning de Goldberg (1989) se presentan los conceptos fundamentales de los algoritmos genéticos y su aplicación en problemas de optimización complejos.

El funcionamiento de un AE comienza con la generación de una población inicial de soluciones potenciales, que suelen representarse como vectores en un espacio de búsqueda. En cada generación, los individuos son evaluados en función de su aptitud (fitness), que refleja su capacidad para resolver el problema planteado. Luego, se seleccionan los individuos más aptos para dar lugar a una nueva generación, que se forma mediante procesos de cruzamiento (recombinación) y mutación. Este ciclo de evaluación y evolución se repite hasta alcanzar un criterio de parada, como un número máximo de generaciones o la estabilización de los resultados.

Aunque los AE no garantizan la obtención de la solución óptima global, su eficacia ha sido demostrada en numerosos contextos. Su comportamiento se respalda en fundamentos teóricos sólidos, provenientes de la teoría evolutiva, la complejidad computacional y estudios sobre la convergencia de algoritmos.

El algoritmo 1 presenta un pseudocódigo genérico que resume el funcionamiento típico de un algoritmo evolutivo aplicado sobre una población P.

Dentro de los AE, los algoritmos genéticos (AG) son una técnica ampliamente utilizada, basada en la teoría darwiniana de la evolución. Utilizan representaciones codificadas de las soluciones (frecuentemente como cadenas de

Algoritmo 1 Algoritmo evolutivo genérico sobre una población P

```
1: Inicializar(P(0))
2: generación = 0
3: while no se cumple el criterio de parada do
4:     Evaluar(P(generación))
5:     padres = Seleccionar(P(generación))
6:     hijos = OperadoresEvolutivos(padres)
7:     nuevaPoblación = Reemplazar(hijos, P(generación))
8:     generación++
9:     P(generación) = nuevaPoblación
10: end while
11: return mejor solución encontrada
```

bits o enteros), aplican operadores genéticos como selección, cruzamiento y mutación, y emplean una función de aptitud para guiar la búsqueda hacia soluciones cada vez mejores.

Los AG, y en general los AE, han demostrado ser herramientas valiosas para la optimización, especialmente cuando se combinan con formulaciones hidráulicas precisas del problema o con herramientas de simulación especializadas. En el contexto del diseño de redes de saneamiento, permiten abordar simultáneamente múltiples variables y restricciones, optimizando la disposición de las tuberías, la ubicación de los sumideros y el dimensionamiento de componentes clave. El objetivo final es minimizar los costos de construcción y operación, maximizar la eficiencia hidráulica y cumplir con las normativas técnicas y ambientales.

2.1.2. Redes de saneamiento

Antes de definir el problema, es conveniente repasar algunos conceptos básicos que se manejan en el diseño de redes de saneamiento. Estas redes abarcan todo el sistema encargado de evacuar las aguas residuales y pluviales, desde las conexiones domiciliarias o industriales hasta las plantas de tratamiento, donde el agua es tratada antes de ser devuelta al medio ambiente. El funcionamiento se basa, en su mayoría, en el escurrimiento por gravedad, por lo que es fundamental que las tuberías tengan una pendiente suficiente y estén bien diseñadas para evitar obstrucciones. Por lo general, se separan las redes de aguas residuales y las pluviales, ya que esto permite reducir los costos asociados al tratamiento. En zonas donde el terreno es muy llano y no hay pendiente

natural suficiente, es común incorporar estaciones de bombeo para asegurar el desplazamiento del agua a lo largo del sistema.

El caudal, también llamado demanda o flujo, hace referencia a la cantidad de agua que circula por una tubería o canal en un determinado período de tiempo. En el diseño de redes de saneamiento, este valor es fundamental para dimensionar correctamente las tuberías y demás componentes del sistema. Un cálculo adecuado del caudal permite asegurar que la red pueda transportar sin inconvenientes los volúmenes de aguas residuales previstos, evitando desbordes y garantizando un funcionamiento eficiente.

En el diseño de sistemas de saneamiento deben considerarse diversas restricciones técnicas que aseguren el funcionamiento correcto y seguro de la red. A continuación, se presentan algunas de las principales:

- Excavación: Es necesario contemplar las condiciones geográficas y geotécnicas del terreno, así como los costos asociados al movimiento de tierras. Además, suele existir una profundidad mínima de instalación que deben respetar las tuberías, lo cual influye en el trazado de la red y en la ubicación de las estructuras.
- Pendiente: La pendiente de las tuberías es clave para asegurar el flujo por gravedad. Se deben respetar valores mínimos y máximos, ya que una pendiente insuficiente puede generar sedimentación, mientras que una excesiva puede provocar velocidades elevadas que dañen las tuberías o arrastren sólidos de forma ineficiente.
- Capacidad de transporte: Las tuberías deben tener un diámetro adecuado para conducir el caudal esperado sin generar desbordes ni obstrucciones. Para ello, se consideran aspectos como la velocidad del flujo, la capacidad hidráulica del conducto y su resistencia interna.
- Normativas y regulaciones: El diseño debe ajustarse a las normativas locales y nacionales vigentes. Estas regulaciones establecen requisitos vinculados a la calidad del agua, las condiciones de descarga, aspectos ambientales y de seguridad sanitaria, todo con el fin de proteger la salud pública y el entorno.

Capítulo 3

Relevamiento del estado del arte

Previo al abordaje del modelado de soluciones en el contexto del diseño de redes de saneamiento, se considera fundamental realizar una revisión de antecedentes. Esta fase implica el análisis detallado del estado del arte relacionado con la problemática en cuestión, buscando identificar tanto las contribuciones más relevantes como las brechas existentes en la literatura científica. En esta etapa, se llevó a cabo un minucioso relevamiento de una amplia gama de publicaciones especializadas, que abarcan desde investigaciones académicas hasta desarrollos industriales y tecnológicos, con el objetivo de obtener una comprensión integral y actualizada del campo de estudio. La revisión de antecedentes no solo proporciona un marco teórico sólido para fundamentar la investigación, sino que también permite identificar oportunidades de mejora y posibles enfoques innovadores para abordar el problema planteado.

Para la selección de los artículos presentados en el estado del arte, se llevó a cabo un proceso de revisión bibliográfica estructurado. La búsqueda se realizó principalmente a través del Portal Timbó, el cual brinda acceso a las principales colecciones de publicaciones científicas y bases de datos académicas a nivel mundial. Se utilizaron combinaciones de palabras clave en español e inglés, tales como 'diseño de redes de saneamiento', 'optimización de alcantarillado', 'genetic algorithms for sewer design' y 'evolutionary computation in wastewater networks'. Una vez recopilado un conjunto inicial de publicaciones, se aplicaron los siguientes criterios de selección para conformar la lista final de trabajos a analizar en profundidad:

- Relevancia Temática: Se priorizaron aquellos estudios cuya temática principal fuera la aplicación directa de algoritmos evolutivos o genéticos al problema específico del diseño de redes de saneamiento.
- Detalle Metodológico: Se hizo especial foco en artículos que describieran en detalle la formulación del problema de optimización. Fue fundamental que los trabajos detallaran explícitamente la función objetivo y las restricciones utilizadas, ya que esto permitía un análisis comparativo de los diferentes enfoques.

Este proceso de filtrado permitió concentrar el análisis en una selección de artículos que no solo eran pertinentes, sino que ofrecían la profundidad técnica necesaria para construir un marco teórico sólido y comparar las estrategias de optimización existentes.

3.1. Artículos relevados

En esta sección, se presenta una selección de los artículos científicos más relevantes y significativos identificados durante el proceso de revisión de antecedentes. Los trabajos fueron cuidadosamente evaluados y seleccionados por su contribución al conocimiento existente en el campo de los algoritmos evolutivos aplicados al diseño de redes de saneamiento. De los artículos relevados, se hizo foco en ver cómo formularon la optimización del problema, destacando las diferentes definiciones de la función objetivo y las restricciones utilizadas. Esta recopilación de literatura especializada no solo enriquece el marco teórico de la tesis, sino que también sirve como punto de partida para el desarrollo y la validación de los enfoques propuestos.

3.1.1. Optimum layout design of sewer networks by hybrid genetic algorithm (Hassan et al. 2020)

Hassan et al. 2020 presentaron un algoritmo innovador que combina un AG con un enfoque de tree growing algorithm (TGA) para lograr la optimización del diseño de redes de alcantarillado, buscando minimizar los costos y asegurar un rendimiento óptimo del sistema.

El TGA se utilizó inicialmente para generar de forma rápida un conjunto de soluciones primarias en un grafo base. Posteriormente, este grafo se aplicó

como población inicial en el algoritmo genético. Los autores argumentaron que esta estrategia híbrida ofrece ventajas significativas, ya que el AG restringe de manera considerable el espacio de búsqueda en comparación con su uso individual, lo que conduce a una convergencia más rápida hacia una solución.

En este enfoque, cada tubería se asocia con un peso que está relacionado con el costo de construcción. El costo puede depender del diámetro de las tuberías, de la profundidad de excavación y de la presencia de estaciones de bombeo. Los pesos de las tuberías se definen mediante una función cóncava que considera la longitud de las tuberías y el caudal de descarga del sistema de alcantarillado. Dado que la descarga acumulada de las tuberías no es inicialmente conocida, el costo total del sistema se obtiene a través de la configuración implícita del diseño. La ecuación 3.1 describe la función objetivo que utilizaron, donde C representa el costo del diseño objetivo, L denota la longitud de la tubería, Q representa el caudal acumulado de la tubería obtenido en la configuración del diseño, y n es la cantidad total de tuberías. El caudal en cada tubería se calcula como la suma de los caudales de las tuberías anteriores.

$$\text{mín}\, C = \sum_{i=1}^{n} L_i \sqrt{Q_i} \tag{3.1}$$

Además de la función objetivo, se plantearon algunas restricciones básicas que debían cumplirse para generar un diseño factible a partir del gráfico inicial:

1. El gráfico debe ser acíclico.
2. En la configuración del diseño, todos los pozos (vértices) deben estar involucrados en el árbol, no puede haber pozos de registro aislados.
3. Todas las tuberías (aristas) en la configuración del diseño deben formar parte del árbol, ya que cada una de ellas drena aguas residuales para una calle específica.
4. El sistema de drenaje debe tener una salida claramente direccionada por el árbol.
5. Es posible que múltiples tuberías ingresen a un pozo, pero el caudal debe fluir a través de una sola tubería que apunta hacia la salida principal.

Las restricciones aseguran la viabilidad del diseño del sistema de alcantarillado. Todas estas condiciones de la red se cumplen automáticamente en el TGA, lo que resulta en un diseño final factible.

Los autores concluyeron que TGA es un algoritmo con un método efectivo para la formación de las poblaciones iniciales de la solución, ya que no genera árboles aleatorios, lo que garantiza que todas las soluciones sean factibles. Para implementar este algoritmo, utilizaron 3 vectores: A, C, AA, siendo A el conjunto de aristas, C el conjunto de vértices dentro del árbol, y AA un conjunto de aristas vecinas en el árbol donde cualquiera de las cuales tiene la misma probabilidad de formar la siguiente rama. El algoritmo 2 es el que presentaron Hassan et al. (2020) para describir como implementaron el TGA.

Algoritmo 2 Tree-Growing algorithm

1: Inicializar $C = [N_r]$, $A = []$, y $AA =$ [aristas conectadas al nodo raíz].
2: Seleccionar una arista a cualquiera de AA y agregar $A = A + [a]$.
3: Seleccionar el vértice de la arista a como el nuevo vértice N y agregar a $C = C + [N]$.
4: Identificar aristas, $ac(i)$, conectadas a N en el grafo, excluyendo a.
5: Actualizar AA, eliminando la arista a y agregar todas las aristas de $ac(i)$ que sean posibles candidatas.
6: Repetir los pasos 2 a 5 hasta obtener el número correcto de aristas agregadas en el árbol.

3.1.2. A GA-HP Model for the Optimal Design of Sewer Networks (Hassan et al. 2018)

Hassan et al. (2018) buscaron obtener una solución rentable que minimizara la inversión de capital en el diseño de redes de saneamiento, manteniendo un rendimiento eficaz del sistema. Para lograr este objetivo, propusieron un modelo innovador que combina un algoritmo genético con un modelo de programación heurística (GA-HP, por sus siglas en inglés).

En una primera etapa, el AG se utilizó para determinar los diámetros de las tuberías en el diseño preliminar de la red. Luego, mediante programación heurística (HP), se establecieron la pendiente óptima de las tuberías y otras características del sistema, como la velocidad y el costo total.

La función presentada en la ecuación 3.2 corresponde a la función objetivo utilizada, donde C representa el costo total de la red de saneamiento, D_i el diámetro de la tubería en el tramo i, AH_i la profundidad media de excavación para el tramo i, CS_i el costo de la tubería y excavación para dicho tramo, y L_i la longitud de la tubería i.

Para evitar que se consideren soluciones no factibles, los autores propusieron una función de penalización definida en la ecuación 3.3, donde δ representa el parámetro de penalización y g_{ij} la j-ésima violación de la restricción correspondiente al tramo i.

$$\text{mín}\, C = \sum_{i=1}^{N} f_i(D_i, AH_i, CS_i) \times L_i \tag{3.2}$$

$$\text{mín}\, C = \sum_{i=1}^{N} f_i(d_i, \bar{Z}_i, CS_i) \times L_i + \sum_{j=1}^{7} \delta_j \sum_{i=1}^{N} (g_{ij}) \tag{3.3}$$

La función objetivo se encuentra sujeta a ciertas restricciones, principalmente de naturaleza hidráulica y económica. Los autores demostraron que el enfoque de penalización utilizado en el modelo es efectivo para garantizar la satisfacción de las restricciones. Esto significa que, a través de este enfoque, fue posible asegurar que las soluciones propuestas por el modelo no solo eran óptimas desde el punto de vista de la función objetivo, sino que también respetaban las restricciones impuestas. Las restricciones que plantearon fueron las siguientes:

1. $V_{min} \leq V_i \leq V_{max}$
   La velocidad del flujo debe ser mayor o igual que el mínimo permisible para prevenir la sedimentación y menor o igual que el máximo permisible para prevenir la erosión de la tubería.
2. $D_{i-1} \leq D_i$
   El diámetro de cualquier tubería i debe ser mayor o igual que el diámetro de la tubería $i-1$ que desemboca en ella.
3. $S_{min} \leq S_i \leq S_{max}$
   La pendiente de cada tubería debe estar entre un valor permitido, donde S_i es la pendiente de la tubería i, S_{min} es la pendiente mínima permitida y S_{max} es la máxima.
4. $AD_{min} \leq GE_i - UD_i$
   La tubería debe tener una profundidad mínima requerida que depende de factores locales y del material usado. GE_i es la elevación del suelo en el extremo más alto de la tubería (aguas arriba), UD_i es la profundidad aguas arriba (punto más alto) de la tubería, AD_{min} es la profundidad mínima permitida.

5. $GE_i - DD_i \leq AD_{max}$
   La profundidad máxima permitida de la tubería depende de las condiciones del subsuelo, donde DD_i es el punto más bajo de la tubería (aguas abajo) y AD_{max} es la profundidad máxima permitida.

El modelo híbrido GA-HP se desarrolló a través de los siguientes pasos:

1. Codificación de las variables utilizando una codificación de enteros. El largo del cromosoma es la cantidad de tuberías del problema.
2. Generación aleatoria de la población inicial (prueba con varios tamaños desde 30 a 200).
3. Decodificación y evaluación de la población utilizando la programación heurística.
4. Se hace el llamado a la subrutina con el algoritmo HP, que se realiza para cada cromosoma. En este paso se calcula la pendiente, profundidad de colocación y el costo de la red.

   a) Se calcula la pendiente óptima.
   b) Se calculan las características hidráulicas de la red (velocidad, etc) y las restricciones.
   c) Se aplica la función de penalización en caso de que la solución no cumpla las restricciones.
   d) Se calcula el costo total de la red. El costo es obtenido como la suma de los costos de instalación de las tuberías, pozos de registro y la función de penalización.

5. Cálculo del fitness basado en el costo total.
6. Se genera una nueva población.

   a) La selección se hace seleccionando 2 padres de manera probabilística, mejor fitness implica mejor probabilidad. Se puede aplicar selección de torneo, rueda de ruleta y Exponential Rank Selection (ERS).
   b) Cruzamiento: Comenta que se consideraron cuatro métodos distintos de cruzamiento: Single point, uniform, Flat arithmetic crossover, Intermediate crossover. En caso de que no se aplique cruzamiento se colocan los padres tal cual en la nueva población.

c) Mutación: Se selecciona un único gen a mutar con probabilidad 0.5. Es una mutación común, se cambia el entero por otro dentro del rango establecido (lista de diámetros).

7. Se aplican los operadores del paso 6. hasta formar una nueva población.
8. Se repiten todos los pasos hasta la convergencia.

Los autores comentaron que el mejor resultado se obtuvo utilizando selección por torneo, con cruzamiento de un punto. También evidenciaron que el enfoque de un algoritmo genético híbrido con un modelo de programación heurística, ofrece una solución rentable que minimiza la inversión de capital y asegura un buen rendimiento del sistema, donde además supera en resultados a trabajos anteriores, Mays y Yen (1975) y Masnsouri y Khanjani (1999), que utilizaron métodos como la programación lineal y autómatas para resolver problemas similares de diseño de redes de saneamiento.

3.1.3. Optimization of Potable Water Distribution and Wastewater Collection Networks: A Systematic Review and Future Research Directions (Zhao et al. 2016)

El estudio presentado por Zhao et al. (2016) tuvo como objetivo principal minimizar el costo total de construcción de redes de distribución de agua potable y recolección de aguas residuales. Determinaron el costo en función del diámetro, longitud y profundidad de las tuberías, representado por la ecuación 3.4 donde N es el número de tuberías, μ es la función de costo asociada a la tubería, D_i es el diámetro de la tubería i, UD_i es la altura de excavación del vértice superior de la tubería, DD_i es la altura de excavación del vértice inferior, y L_i es la longitud de la tubería i.

$$C = \sum_{i=1}^{N} \mu_i(D_i, UD_i, DD_i, L_i) \tag{3.4}$$

En el artículo se establecieron una serie de restricciones para garantizar la viabilidad de las soluciones, entre las cuales se incluyen:

1. $D_i \in D$
   El diámetro es una variable discreta dentro del conjunto de diámetros disponibles en el mercado.

2. $\phi_i \leq \phi_{max}$; ϕ es el ángulo central
3. $Q_i = f_i(D_i, \phi_i, S_i)$
   Donde Q_i es el flujo de la tubería i (m^3/s), S_i es la pendiente.
4. $V_{min} \leq V_i \leq V_{max}$
   Donde V_i hace referencia a la velocidad del flujo en la tubería i
5. $V_i = Q_i/A_i$ Donde A_i es el área del flujo en un corte transversal en la tubería i
6. $D_{i-1} \leq D_i$
7. $AD_{min} \leq UD_i \leq AD_{max}$
   Donde AD_{min} y AD_{max} hacen referencia a la profundidad de excavación mínima y máxima permitida y UD_i es la profundidad a la que se encuentra el extremo aguas arriba de la tubería i
8. $AD_{min} \leq DD_i \leq AD_{max}$
   DD_i es la profundidad a la que se encuentra el extremo aguas abajo de la tubería i
9. $GE_i - UD_i \leq GE_i - DD_{i-1}$
   GE_i es la altura del terreno
10. $GE_i - UD_i + D_i \leq GE_i - DD_{i-1} + D_{i+1}$
11. $(GE_i - UD_i - (GE_{i+1} - DD_i))/L_i \geq S_{min}$

Fueron analizados diversos enfoques y algoritmos utilizados en la optimización de redes de saneamiento, entre los métodos deterministas como la programación dinámica y la programación lineal, los autores evidenciaron limitaciones para encontrar óptimos en espacios no lineales o con gran cantidad de nodos.

Al hablar de meta-heurísticas, Zhao et al. (2016) dividieron el análisis entre las meta-heurísticas basadas en población y las que no son basadas en población. Destacaron la efectividad de los algoritmos genéticos como una técnica de optimización basada en población.

También comentaron sobre trabajos que utilizaron una combinación de AG con simuladores hidráulicos de aguas residuales, como el módulo TRANSPORT de SWMM, un software desarrollado por la Agencia de Protección Ambiental de Estados Unidos (EPA), para el modelo de gestión de aguas pluviales. Se mencionan dos modelos, el primero GA-TRANS1 en el cual utiliza el AG para hallar el diámetro y la elevación de las tuberías, mientras que TRANSPORT realiza el análisis hidráulico. El segundo, GA-TRANS2 usa los AG para hallar

únicamente la elevación, y tanto los diámetros como el análisis hidráulico es realizado por TRANSPORT.

Los autores relevaron el trabajo de Haghighi y Bakhshipour (2012) en el cual se menciona la dificultad para hallar buenas soluciones en caso de tener muchas restricciones, donde también se desarrolló la estrategia adaptive sequential constraint.

El estudio también abordó los métodos basados en Ant Colony Optimization (ACO). Estos métodos son originalmente utilizados para problemas combinatorios utilizando variables de decisión discretas. En el caso de redes de recolección de aguas residuales, se cuentan con variables continuas (profundidad de excavación, pendiente, etc) por lo cual se requiere discretizar los valores.

Una combinación entre el enfoque Cellular Automata for Sewers in Network Optimization (CASiNO) y NSGA-II fue utilizada para comparar los resultados del enfoque multiobjetivo. CASiNO es utilizado para proveer a NSGA-II de buenas soluciones iniciales con el objetivo de acelerar la optimización. Los objetivos tomados fueron la reducción del capital gastado y la minimización de inundaciones. Usaron también el SWMM como simulador hidráulico para evaluar las soluciones generadas.

3.1.4. Designing Wastewater Collection Systems Using Genetic Algorithms (Liang et al. 2000)

Liang et al. (2000) propusieron el uso de algoritmos genéticos para configurar tuberías en una red de saneamiento. El enfoque consideró criterios hidráulicos realistas para definir restricciones como el diámetro, velocidad mínima/máxima, cubrimiento mínimo/máximo y nivel invertido, muy similares a las relevadas en otros artículos. Las restricciones que se plantearon fueron:

1. $V_{min} \leq V_i \leq V_{max}$
   La velocidad del flujo se deben encontrar en el rango permitido
2. $D_{i-1} \leq D_i$
   Los diámetros deben ser mayores o iguales a los diámetros utilizados aguas arriba
3. $AD_{min} \leq UD_i \leq AD_{max}$
   La profundidad del extremo aguas arriba de la tubería debe encontrarse en el rango permitido

4. $AD_{min} \leq DD_i \leq AD_{max}$
   La profundidad del extremo aguas abajo de la tubería debe encontrarse en el rango permitido

Para el algoritmo, utilizaron una codificación binaria y una función de penalización para los diseños candidatos que no satisfacen la restricción del diámetro de progresión. La función de fitness es una suma lineal de tres costos: el costo de materiales de las tuberías, el costo de construcción (excavación, relleno y vertido) y el costo de penalización para diseños de red que no cumplen con las restricciones. En tanto, la función de penalización la definieron por la ecuación 3.5 donde PF es un factor de penalización que definieron con el valor 200, y D_i corresponde al diámetro de la tubería i.

$$Penalty = PF \times (D_i - D_{i+1}) \tag{3.5}$$

El modelo fue evaluado en una red con solo 18 enlaces del proyecto East-I Wastewater Collection Engineering, ubicado en el Parque Industrial Costero Changbin, en China. En comparación con métodos tradicionales, el algoritmo demostró ser efectivo al minimizar costos y proporcionar un diseño eficiente. Posteriormente, en 2004, los autores ampliaron su trabajo incorporando la búsqueda por tabú (Tabu Search, TS) para desarrollar un operador de búsqueda mejorado en torno a una solución candidata. Esta vez, la evaluación incluyó la red completa con 105 enlaces como caso de estudio. El enfoque evolutivo que propusieron logró ahorros de costos del 9 % (utilizando solo la optimización AG) y hasta el 16 % al combinar AG y TS, en comparación con el diseño de costos reales.

3.1.5. Optimization of Sewer Networks Using an Adaptive Genetic Algorithm, (Haghighi y Bakhshipour, 2012)

Haghighi y Bakhshipour (2012) presentaron un enfoque de optimización de redes de alcantarillado utilizando un algoritmo genético adaptativo. El objetivo principal fue encontrar el diseño óptimo de la red de alcantarillado que cumpla con las restricciones técnicas y minimice los costos de construcción y operación.

Con la función objetivo definida en la ecuación 3.6 buscaron la minimización de los costos totales de construcción y operación de la red de alcantarillado, que incluían los costos de excavación, instalación de tuberías y bombas, y los costos de operación y mantenimiento. Donde C es la función de costo, $CS =$ es el costo de construcción de alcantarillas y $CM =$ costo construcción del pozo de registro. Ambos costos son en función del diámetro de la tubería utilizada y la profundidad a la que se entierra. Cpu es el costo de construcción de las estaciones de bombeo, φ_i indica que hay ($\varphi_i = 1$) o no hay ($\varphi_i = 0$) una estación de bombeo en la tubería i y N es el número de tuberías. En este trabajo se cumple que $\varphi_i = 0$ ya que no se implementaron estaciones de bombeo.

$$C(D, S, P) = \sum_{i=1}^{N} (CS_i + \varphi_i \times Cpu_i) + \sum_{i=1}^{N+1} CM_i \tag{3.6}$$

Las restricciones planteadas incluyeron la capacidad de las tuberías para transportar el caudal de diseño, la velocidad del flujo dentro de las tuberías, las pendientes mínimas y máximas permitidas, y la profundidad mínima de cobertura de las tuberías. Las restricciones fueron formuladas utilizando ecuaciones que relacionan los parámetros de diseño con las variables de decisión del algoritmo genético. Dejamos a continuación las restricciones que utilizaron:

1. Mantener la velocidad del flujo entre el valor máximo y mínimo. La velocidad puede ser calculada por medio de la fórmula de Manning
2. La tubería i debe estar por encima de la tubería $i + 1$
3. Mantener la proporción de la profundidad de las aguas
4. Elegir diámetros de tuberías de los elegibles comercialmente
5. Mantener una mínima profundidad de excavación
6. El diámetro de la tubería $i + 1$ debe ser mayor o igual al diámetro de la tubería i

El enfoque propuesto consta de dos etapas: en la primera etapa, los cromosomas binarios se decodificaron en parámetros de diseño normales, como diámetros de tubería y pendientes. En la segunda etapa, los parámetros de diseño normales se convirtieron en diseños factibles, considerando las restricciones técnicas y ajustando las elevaciones de instalación de las tuberías.

En resumen, el artículo propuso un enfoque de optimización de redes de alcantarillado utilizando un algoritmo genético adaptativo, con el objetivo de minimizar los costos totales y cumplir con las restricciones técnicas.

3.1.6. GA-GHCA model for the optimal design of pumped sewer networks (Rohani y Afshar, 2015)

Rohani y Afshar (2015) presentaron un modelo híbrido que combina un algoritmo genético y un autómata celular híbrido general (GHCA) para el diseño óptimo de redes de alcantarillado. El problema de diseño de la red se formuló como la minimización de costos totales, que incluyeron el costo de instalación de tuberías, pozos de registro, pendientes y operación de bombas. La fórmula matemática con la que plantearon el problema para una red fija es la definida en la ecuación 3.7. Los componentes de costo de esta ecuación se detallan con mayor especificidad en la ecuación 3.8

$$\min C = \sum_{i=1}^{N} CS_i + \sum_{i=1}^{NN} CM_i + \sum_{i=1}^{NN} Cd_i + \sum_{i=1}^{NN} Cpu_i \tag{3.7}$$

$$\min C = \sum_{i=1}^{N} L_i K_p(D_i, UD_i, DD_i) + \sum_{i=1}^{NN} Km(hm_i) + \sum_{i=1}^{NN} kd(hd_i) + \sum_{i=1}^{NN} Cpu_i \tag{3.8}$$

A continuación se describen las variables definidas en las ecuaciones 3.7 y 3.8:

- C es el costo de la red,
- CS_i es el costo de instalación de la tubería i,
- CM_i es el costo del pozo de registro i,
- Cd_i es el costo de la pendiente instalada si es necesaria,
- Cpu_i representa los costos de la instalación y operación de bombas en el nodo i,
- N es el número de tuberías en la red,
- NN es el número de nodos en la red,
- L_i es el largo de la tubería i,
- D_i es el diámetro de la tubería i,
- UD_i, DD_i son las profundidades del punto aguas arriba y aguas abajo de la tubería i,

- hm_i es la profundidad del pozo de registro i,
- hd_i es la altura de la pendiente i,
- Kp es el costo unitario de la instalación de la tubería i, en función de su diámetro D_i y la profundidad (UD_i, DD_i),
- Km es el costo de construcción del pozo en función de la profundidad (hm),
- Kd es el costo de construcción de la pendiente en función de la pendiente (hd).

Las restricciones incluyen la velocidad del flujo, la profundidad relativa del flujo, la pendiente, la profundidad mínima y máxima de las tuberías, así como la discreción de los diámetros disponibles. Estas restricciones se expresan matemáticamente y están relacionadas con parámetros como el diámetro, las profundidades aguas arriba y aguas abajo, la profundidad de los pozos y la altura de la pendiente.

1. $V_{min} \leq V_i \leq V_{max}$
   La velocidad del flujo debe estar dentro de valores mínimos y máximos permitidos.
2. $\beta_{min} \leq \beta_i \leq \beta_{max}$
   Relaciona la profundidad del agua, donde $\beta_i = \frac{y_i}{D_i}$ es la profundidad relativa del flujo en la tubería l, y_i es la profundidad del flujo y D es el conjunto de diámetros de tuberías disponibles.
3. $S_{min} \leq S_i \leq S_{max}$
   La pendiente debe estar dentro de los valores permitidos.
4. $AD_{min} \leq UD_i, DD_i \leq AD_{max}$
5. $D_i \in D$
   El diámetro es una variable discreta dentro del conjunto de diámetros disponibles en el mercado.
6. $D_i \leq D_{i+1}$ El diámetro de la tubería debe ser menor o igual que la que le sigue aguas abajo

La novedad del artículo radica en que la ubicación y la altura de las estaciones de bombeo se abordaron explícitamente con el AG, mientras que el resto del diseño se realizó con el GHCA. Presentaron dos métodos híbridos: en el primero, GA-GHCA1, donde la ubicación y altura de las bombas fueron

sugeridas por el AG, y el resto del diseño se calculó con el GHCA. En el segundo, GA-GHCA2, solo la ubicación de las bombas fue determinada por el AG, dejando el resto del diseño al GHCA.

Los términos claves incluyeron costos de instalación, operación de tuberías y bombas, restricciones de flujo, diámetros de tuberías, profundidades y métodos híbridos de AG y GHCA.

3.1.7. Establishing an Optimization Model for Sewer System Layout with Applied Genetic Algorithm (Weng y Liaw, 2015)

El artículo de Weng y Liaw (2015), propone un modelo de optimización para el diseño de sistemas de alcantarillado que se compuso mediante dos fases: primero, seleccionar el diseño de la red y luego determinar el resto de las propiedades de la red como los tamaños de las tuberías, la profundidad, etc.

La función objetivo descrita en la ecuación 3.9 fue propuesta por los autores para minimizar el costo total del sistema, denotado como C, el cual contempla la suma de los costos asociados a las tuberías, los pozos de registro y las estaciones de bombeo. Las variables de decisión incluyen el diámetro y la pendiente de las tuberías, la profundidad aguas arriba y aguas abajo, así como también las características del sistema de bombeo.

$$\text{mín}\, C = \sum_{i=1}^{N} \sum_{j=1}^{M} \text{Cost}(D_{ij}, S_{ij}, UD_i) \cdot X_{ij} + \sum_{i=1}^{N} \text{CM}(UD_i) + \sum_{i=1}^{N} \text{PS}_i(Q_i, H_i) \quad (3.9)$$

A continuación se definen las variables utilizadas:

- D_{ij}: diámetro de la j-ésima opción para la i-ésima tubería
- S_{ij}: pendiente asociada al diámetro D_{ij}
- UD_i: profundidad aguas arriba del tramo i
- DD_i: profundidad aguas abajo del tramo i
- X_{ij}: variable binaria que indica si se selecciona la opción j para el tramo i
- H_i: altura de bombeo necesaria (pumping head) en el punto i
- Q_i: caudal de bombeo en el punto i
- $CM(UD_i)$: costo del pozo de registro en función de la profundidad

- $P_i(Q_i, H_i)$: costo de la estación de bombeo asociada al punto i
- $AD_{\min}$ y $AD_{\max}$: profundidades mínima y máxima permitidas para la instalación de tuberías

Las restricciones utilizadas buscan asegurar la coherencia hidráulica y constructiva del sistema. Incluyen la selección discreta de opciones de tuberías mediante variables binarias, límites mínimos y máximos para las profundidades de instalación, y una relación de no decrecimiento entre los diámetros de tuberías consecutivas:

1. $X_{ij} = 0, 1$, donde X_{ij} indica si se selecciona la j-ésima opción de la i-ésima tubería
2. $\sum_{j=1}^{M} X_{ij} = 1$, para garantizar que cada tubería tenga exactamente una configuración seleccionada
3. $AD_{\min} \leq UD_i \leq AD_{\max}$, restricción sobre la profundidad aguas arriba
4. $AD_{\min} \leq DD_i \leq AD_{\max}$, restricción sobre la profundidad aguas abajo
5. $D_{i-1,j} \leq D_{i,j}$, para asegurar que el diámetro no disminuya entre tramos consecutivos

El objetivo principal fue minimizar el costo total de construcción y operación de la red de alcantarillado, lo que les implicó seleccionar tamaños de tuberías, pendientes y estaciones de bombeo que cumplieran con las restricciones y minimicen el costo general del diseño.

3.2. Comentarios generales sobre los estudios

Los trabajos revisados han utilizado funciones de costo simples que se relacionan linealmente con la longitud de las tuberías de alcantarillado y con la raíz cuadrada del caudal acumulado en las tuberías. También han incorporado restricciones importantes para asegurar un diseño del árbol de expansión que tenga en cuenta todas las alcantarillas y la salida del sistema de drenaje. Otra restricción presente en los artículos revisados es que se permite que varias tuberías de alcantarillado desemboquen en una boca de acceso, pero se limita a una sola tubería la salida de esa boca.

Además, los artículos revisados abordan un segundo subproblema de optimizar el diseño hidráulico, que considera la selección, ubicación y configuración de los elementos de la red en una distribución preestablecida. En este caso, el

diseño de la red se resuelve por separado utilizando enfoques heurísticos simples o aproximados, donde se aplicaron métodos evolutivos para la selección y configuración de los elementos. Si bien esta estrategia permite determinar las propiedades físicas adecuadas para la red, está limitada por el diseño subyacente establecido previamente.

La mayoría de los estudios revisados coinciden en la formulación de la función objetivo y las restricciones. Aunque las funciones objetivo varían ligeramente entre sí, todas convergen en la búsqueda por minimizar los costos de construcción. Estos costos se definen considerando variables tales como la longitud y/o diámetro de las tuberías, la profundidad y, en algunos casos, la necesidad de estaciones de bombeo. Respecto a las restricciones, existe una consistencia notable entre los distintos trabajos. El diámetro de las tuberías se considera como una variable discreta dentro del conjunto de diámetros disponibles en el mercado. Además, se establece que el diámetro de una tubería debe ser menor o igual al de la tubería siguiente aguas abajo. También se imponen límites de velocidad para el flujo y, en algunos casos, restricciones sobre pendientes y profundidades de las tuberías.

Capítulo 4

Metodología

En este capítulo se presenta la metodología desarrollada para abordar el problema de diseño de redes de saneamiento. Primero se detalla el proceso de construcción del escenario utilizado, incluyendo la descripción, representación y los pasos necesarios para su generación a partir de datos geoespaciales. Luego se describen los algoritmos implementados para obtener soluciones al problema, un algoritmo de tipo greedy y un algoritmo evolutivo.

4.1. Armado del escenario

En esta sección se describe el proceso realizado para la construcción del escenario utilizado en el proyecto.

4.1.1. Descripción del escenario

La realidad del problema a abordar se basa en la construcción de redes de saneamiento en ciudades de interés. La red es compuesta por pozos y cañerías. Por simplicidad y para reducir costos de construcción, al evitar la necesidad de adquirir terrenos privados, se decidió que el trazado de la red respete el trazado de calles de la ciudad, de esta forma las cañerías siguen el trazado de las calles y los pozos se ubican en las esquinas formadas por las intersecciones de las calles. Por lo tanto, el escenario a representar es el trazado existente compuesto de esquinas y calles de la ciudad.

Al describir el problema y relevar estudios previos en el área, se identificaron tres parámetros importantes que se tomaron en cuenta para poder realizar el estudio de manera correcta. Los parámetros son la altura a la que se encuentran

las esquinas, la población total de cada manzana y la distancia entre cada una de las esquinas. A continuación, se detallan los tres parámetros y su relevancia para el armado del escenario.

Por el modelo de red que se desea construir, uno de los puntos importantes de la realidad es el dato de la altura a la que se encuentra cada intersección con respecto a las demás intersecciones. La importancia de este dato radica en que las redes a diseñar son gravitacionales, lo cual implica que el flujo de las aguas servidas es propulsado únicamente por la gravedad y la fuerza de arrastre que esta genere, sin necesidad de contar con estaciones de bombeo en la red. Por lo tanto, para representar la necesidad se añade al escenario la altura con respecto al nivel del mar de cada una de las esquinas. Conocer la distancia entre las esquinas es de utilidad para decidir el largo de las cañerías a utilizar en el trazado y es de vital importancia para calcular la pendiente necesaria para cumplir las restricciones hidráulicas. Se entiende por manzana, como el espacio delimitado en todos sus lados por calles. La población residente en cada manzana es de utilidad para calcular el caudal que cada zona de la red debe tolerar.

4.1.2. Representación del escenario

Para lograr una representación computacional del escenario, se optó por emplear grafos como estructura principal para su modelado. Los grafos permiten representar de manera precisa el diseño de la ciudad y a su vez es posible agregar atributos a los elementos que lo componen. En el escenario construido, los vértices del grafo son los encargados de representar a las esquinas y las aristas representan a las calles que conectan estas esquinas, a su vez, la altura se agrega como atributo a los vértices, y la distancia se agrega como atributo a las aristas. Para representar la población de cada manzana en el grafo, se divide la población total de cada manzana entre las calles que la componen, y se le asigna a cada calle la fracción de la población. De este modo, si una calle participa en dos manzanas, la población que debe atender es la suma de las fracciones de población de ambas manzanas.

Para representar el escenario se analizaron diversos formatos para el manejo de grafos, y se utilizó el formato GEXF (Graph Exchange XML Format, 2022). GEXF es un formato que permite describir estructuras complejas de grafos. Uno de los factores clave que influyeron en su elección fue la facilidad

con la que se pueden describir los atributos relacionados con los elementos de la red y sus conexiones. Dado que las representaciones GEXF están escritas en formato XML, es posible visualizarlas con cualquier editor de texto. Otro punto importante es la facilidad con la que se leen y se trabajan con estos tipos de archivos desde diversos lenguajes de programación, ya que existen varias implementaciones de libre acceso. En la subsección 4.1.3 se describen las bibliotecas utilizadas para leer y manipular los grafos.

El escenario se define formalmente como un grafo no dirigido representado en formato GEXF, definido de la forma $G = (V, E)$ donde V es el conjunto de esquinas y E el conjunto de calles. Además, la capacidad del formato GEXF para agregar atributos a vértices y aristas permite representar la realidad de manera completa incluyendo los atributos de altura, longitud de las calles y población.

4.1.3. Construcción del escenario

A grandes rasgos, se divide el proceso de creación del escenario en dos etapas principales:

1. Obtención de los datos geoespaciales requeridos para la construcción
2. Construcción del grafo representativo del escenario utilizando los datos obtenidos.

Entre los datos necesarios para la construcción del escenario se encuentran el trazado de las calles de la ciudad, la altura a la cual se ubican cada una de sus esquinas y la población residente en cada una de las manzanas de la ciudad. Para obtener estos datos, se realizó la búsqueda de la información espacial en portales gubernamentales y no gubernamentales que brindan los datos de forma libre a la población, encontrándose finalmente todos los datos requeridos, en el Sistema de información geográfica de la Intendencia de Montevideo. En este sistema se obtuvieron tres capas con datos geoespaciales, las cuales se describen en la tabla 4.1.

Una vez obtenidos los datos geoespaciales, se comenzó con la construcción del escenario. Para poder trabajar con las capas obtenidas, se utilizó el software QGIS (QGIS Development Team, 2009). Este software permite crear y manipular información geográfica. El primer paso para la construcción del escenario es determinar la zona de interés sobre la cual se desea desarrollar la

Capa	*Descripción*
v_vias	Shapefile con geometría de tipo Line que diagrama el trazado de las calles de la zona a estudiar
v_altura	Shapefile con geometría de tipo MultiLineString que representa las curvas de nivel con precisión de dos metros en la zona a estudiar
v_población	Shapefile con geometría de tipo Polígono representando las manzanas de la zona de estudio, los cuales contienen la cantidad de habitantes de cada una

Tabla 4.1: Descripción de los datos geoespaciales utilizados

solución. Para delimitar esta zona, partiendo de la capa *v_vias* se crea una capa virtual sobre la cual se traza un polígono que se utiliza para delimitar el área de trabajo. Luego, utilizando la herramienta *clip* proporcionada por QGIS se recorta la capa *v_vias* utilizando el polígono definido. De esta manera, se obtienen los datos geoespaciales necesarios para delimitar la zona de interés para el análisis.

El siguiente paso es la creación del grafo incluyendo en él todos los atributos comentados anteriormente. Nuevamente es posible dividir la etapa de representación del escenario en dos etapas separadas.

1. Generación inicial del grafo, incluyendo atributos de población y largo de cada calle
2. Procesamiento de la capa *v_altura* y generación de los atributos en el grafo con la información generada

En la primera etapa se utilizó la biblioteca *momepy* (Fleischmann, 2019) para transformar el GeodataFrame de la capa *v_vias* leído con *GeoPandas* (Jordahl et al. 2022) en una estructura de grafo perteneciente a la biblioteca *NetworkX* (Hagberg et al. 2008). El grafo resultante es un grafo no dirigido sin ningún atributo asignado a sus componentes aún. El grafo resultante refleja con precisión el trazado de las calles de la ciudad que define el caso de estudio. Sin embargo, a pesar de que el grafo generado cumple con los objetivos iniciales de esta sección, es esencial depurarlo para eliminar errores e información redundante generada en el proceso de construcción. Esta depuración abordó principalmente tres tipos de problemas detectados en el grafo inicial:

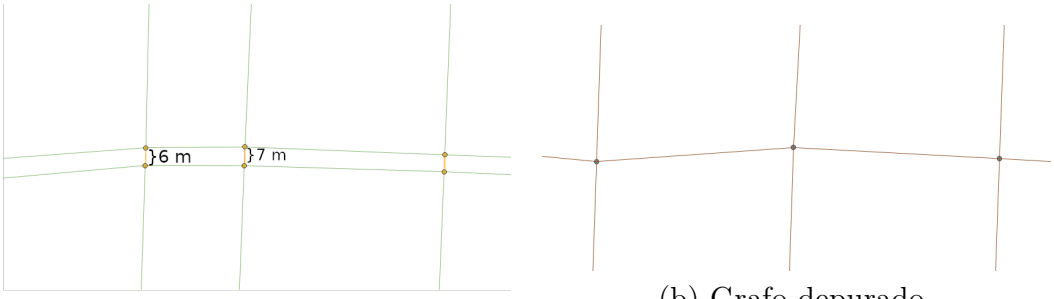
1. Múltiples vértices representando la misma esquina.

2. Lazos en vértices
3. Aristas innecesariamente cortas conectando vértices distintos
4. Eliminar rotondas

El primer tipo de problema consiste en la presencia de múltiples vértices representando la misma esquina, lo cual puede ocurrir por imperfecciones en la capa *v_vias*. Estos vértices duplicados se encuentran a escasos centímetros uno del otro. La corrección implica generar una lista $L_{cercanos} = [(u, v), u \in V, v \in V, d(u, v) < \epsilon]$ compuesta por tuplas de vértices cuya distancia sea menor a un margen ϵ prefijado. En este proyecto se definió el margen en 15 centímetros. Una vez generada la lista $L_{cercanos}$, para cada tupla se aplicó la función *contracted_nodes* provista en *NetworkX*. Esta función, dado un grafo $G = (V, E)$ y dos vértices $(u, v) \in V$, retorna un nuevo grafo $G' = (V', E')$ del cual ha eliminado el vértice v y todas las aristas adyacentes a v se asocian a u. Luego, se eliminaron los lazos (aristas que conectan un vértice consigo mismo) generados por ruido en los datos geoespaciales, pues no representan conexiones urbanas reales. Los lazos se eliminaron utilizando en conjunto las funciones *remove_edges_from* y *selfloop_edges* provistas por *NetworkX*.

En la figura 4.1(a) se ilustra un ejemplo del tercer tipo de problema abordado, aristas cortas conectando vértices. En la representación se muestran vértices conectados por aristas con distancias de 6 y 7 metros. En tales casos, no es necesario incluir una conexión directa entre los vértices en el escenario. Para abordar este problema, se implementó un algoritmo que recorre todos los vértices del grafo y calcula las distancias entre el vértice actual (n) y sus vecinos. Para aquellos vecinos cuyas distancias sean inferiores a un valor predefinido se elimina el nodo vecino y se reconectan sus vértices adyacentes con el vértice n. Para la implementación de este algoritmo el valor establecido para la distancia fue de 50 metros. La figura 4.1(b) muestra cómo la misma sección de la ciudad se representa después de aplicar el algoritmo de depuración al grafo.

Otra depuración realizada implicó la supresión de algunas rotondas en el grafo. El grafo resultante contenía ciclos con áreas pequeñas. Las rotondas estaban compuestas por muchos vértices conectados por aristas de corta distancia, como se detalla en la figura 4.2. Tanto las rotondas como las manzanas se representan como ciclos en el grafo. Para eliminar las rotondas sin afectar las manzanas, se utilizó el área como criterio diferenciador. Se estableció un



(a) Grafo sin depurar

(b) Grafo depurado

Figura 4.1: Comparación del grafo antes (a) y después (b) de la depuración.

umbral de 625m^2, valor que separa adecuadamente las rotondas (generalmente menores) de las manzanas (generalmente mayores). Este umbral es específico para el caso de estudio, ya que el tamaño de las rotondas y manzanas varía significativamente entre ciudades. Se eliminaron todos los ciclos cuya área fuera inferior al umbral definido. Para abordar la eliminación de las rotondas, se definió un punto central para cada una, que pasó a ser el vértice representativo. Luego, se conectaron todas las aristas adyacentes a los vértices originales de la rotonda con este nuevo vértice central. A modo de resumen, la secuencia completa de pasos para la creación y depuración del grafo fue la siguiente:

1. Leer la capa de vías desde el archivo shapefile (‘geopandas.leer_capa_vias‘).
2. Generar la estructura inicial del grafo a partir de las vías (‘momepy.generate_graph‘).
3. Corregir la posición de nodos que representan la misma esquina y están muy cercanos (‘corregir_distancia_nodos‘).
4. Eliminar aristas que conectan un nodo consigo mismo (lazos) (‘eliminar_aristas_lazo‘).
5. Unir vértices distintos conectados por aristas innecesariamente cortas (‘eliminar_vertices_cercanos‘).
6. Eliminar rotondas con area por debajo del umbral definido (‘eliminar_rotondas‘).

Una vez generada y depurada la estructura básica del grafo, se pasa a la generación de los atributos relevantes al problema. El algoritmo 3 detalla el proceso utilizado para el cálculo de la población y distancia entre esquinas para agregarlas como atributos a cada arista del grafo.

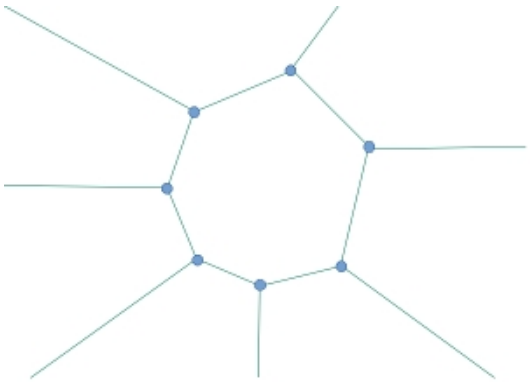


Figura 4.2: Rotonda en grafo

Algoritmo 3 Cálculo de población y distancia

1: Dados u, v
2: distancia = obtener distancia entre las coordenadas u y v
3: población = obtener la población correspondiente a la arista (u,v)
4: asignar atributos población y distancia a la arista (u,v)

En el algoritmo 4 se presenta de manera detallada lo relativo al paso 3 del algoritmo 3, el cual sirve para el cálculo de la población a asignar a cada arista del grafo. Como se mencionó anteriormente, es necesario dividir la población entre todas las calles que componen a la manzana. De forma general, el algoritmo primero determina las calles que componen cada manzana para calcular la población correspondiente y, posteriormente, asigna a cada arista del grafo la suma total de la población que le corresponde de las manzanas adyacentes. Para la implementación se utilizaron los diccionarios *diccionario_zonas* y *poblacion_por_aristas*. En el primero, se almacenaron las aristas más cercanas a cada zona, en el segundo se almacenó la población total que cada arista debe tolerar. En la línea 13 del algoritmo 4, se realiza la iteración para dividir las poblaciones entre todas las aristas a incluir en el escenario. De esta manera, se obtiene la estructura básica del grafo deseado, el cual anteriormente se había definido como el objetivo del paso 1 en la generación de la representación del escenario.

Para agregar al escenario los datos de altura en cada uno de los vértices se utilizó la capa *v_altura*. A diferencia de las restantes capas, *v_altura* se procesó con el software QGIS, debido a su formato particular y a las operaciones de procesamiento requeridas. La estrategia consistió en cargar el grafo resultante de los pasos anteriores y la capa *v_altura* en QGIS para luego utilizar la

Algoritmo 4 Cálculo de población de una arista

Entrada: Un grafo $G = (V, E)$ y los datos de población *v_poblacion*.

```
1:  function ObtenerPoblacion(G, v_poblacion)
2:      dic_zonas = {}
3:      pob_aristas = {}
4:      for (u,v) in V do
5:          centroide = obtenerCentroide(Linea(u,v))
6:          poligonos = zonasCercanas(centroide, v_poblacion, max = 15)
7:          for p in poligonos do
8:              dic_zonas[p] = (u,v)
9:          end for
10:     end for
11:
12:     for zona in dic_zonas do
13:         for arista in dic_zonas[zona] do
14:             if arista not in pob_aristas then
15:                 pob_aristas[arista] = 0
16:             end if
17:             sumar a pob_aristas[arista] la población de la nueva zona
18:         end for
19:     end for
20:     return pob_aristas
21: end function
```

herramienta *join attributes by location*. La herramienta recibe como entrada dos capas y proyecta sobre la primera todos los atributos de la segunda que coincidan en las mismas coordenadas. En la tabla 4.1 se muestra que la capa *v_altura* está compuesta por múltiples líneas con precisión de 2 metros. Por lo tanto, para poder proyectar la altura sobre las coordenadas de los vértices fue necesario crear una superficie continua en el espacio, para lo cual se utilizó un tipo de capa denominada raster. Una capa de datos raster es visualizada como una matriz en el espacio, donde cada celda representa el valor del atributo modelado para esa sección del terreno. Esta estructura modela de forma más natural el mapa de elevación que si fuesen utilizados los datos vectoriales puros. Para construir el raster, se realizó una interpolación entre las distintas curvas de nivel existentes en el área de trabajo. El software QGIS proporciona dos tipos de interpolaciones distintas: Inverse Distance Weighting (IDW) y Triangulated Irregular Network (TIN). Se eligió esta última interpolación debido a ser la más utilizada en problemas de moldeado de mapas de altura (*Interpolating Point Data*, 2024).

Una vez generada la capa raster utilizando la interpolación TIN, se cargó el grafo generado en pasos previos en QGIS. Esta transformación entre estructura de grafo y formato shapefile se realizó utilizando la librería NetworkX, la cual posee la función *write_shp*. El siguiente paso fue aplicar la herramienta *join attributes by location* para asignar los atributos de altura a cada uno de los vértices del grafo. Finalmente, como la representación elegida para el escenario fue el formato GEXF, se utilizó nuevamente NetworkX para leer los shapefiles generados con la altura y exportar el grafo al formato GEXF mediante la función *write_gexf*. En la figura 4.3 se presenta un ejemplo de escenario construido, en el cual el nodo de color rojo hace referencia al nodo sumidero de la red.

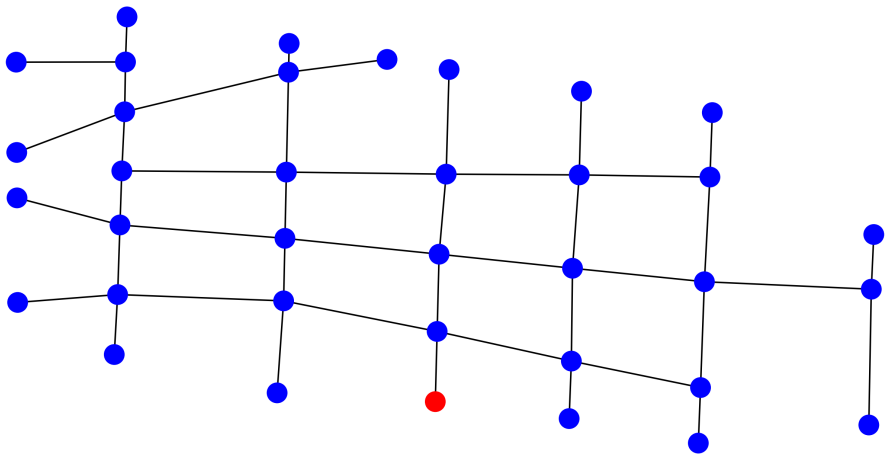


Figura 4.3: Representación del diseño del escenario

En resumen, en esta sección se detalló el proceso de recolección de datos y configuración necesaria para la construcción del escenario a utilizar en la solución, se detallaron los parámetros utilizados y las transformaciones realizadas. En la siguiente sección, se profundiza en la construcción de un algoritmo de tipo greedy y un algoritmo evolutivo, los cuales son capaces de obtener soluciones al problema planteado.

4.2. Algoritmos para resolver el problema

En esta sección se describen dos algoritmos con los cuales se obtienen soluciones al problema planteado. Se describen las implementaciones de un algoritmo de tipo greedy y un algoritmo evolutivo. Si bien los algoritmos de tipo

greedy no ofrecen siempre una solución óptima, este fue implementado por ser una manera intuitiva de resolver el problema y por su capacidad para obtener soluciones rápidamente. Se detallan los pasos requeridos para obtener una solución al problema y se presentan las soluciones intermedias que se obtuvieron en el proceso de construcción de la solución final.

Luego se detalla el algoritmo evolutivo, el cual permite mejorar la solución obtenida mediante el greedy. Se detallan los operadores evolutivos implementados para el algoritmo así como también se comentan decisiones a nivel de diseño tomadas. En el capítulo 5 se presentan los resultados obtenidos aplicando el algoritmo evolutivo a diferentes escenarios creados y se compara los resultados contra los obtenidos mediante el algoritmo greedy.

4.2.1. Algoritmo greedy

Para obtener una base contra la cual comparar el resultado obtenido por medio del algoritmo evolutivo, se implementó un algoritmo de tipo *greedy*. Un algoritmo greedy es una estrategia para la búsqueda de soluciones a problemas en la cual, en cada paso se elige la opción localmente óptima con el objetivo de que estas decisiones óptimas locales lleven a una solución óptima global. Sin embargo, en muchos problemas un algoritmo greedy no garantiza una solución globalmente óptima. Estos tipos de algoritmos son especialmente útiles cuando se necesita obtener una solución rápida a un problema complejo, debido principalmente a que por su naturaleza, resulta sumamente intuitiva su implementación.

4.2.1.1. Descripción del algoritmo de tipo greedy

El proceso de construcción del algoritmo de tipo greedy consta de dos fases principales. En la primera se construye la topología de la red generando un grafo dirigido y acíclico que garantiza el flujo gravitacional. Para esto, se itera sobre cada nodo del escenario, exceptuando al nodo sumidero, y se lo conecta con aquel vecino que se encuentre a una altura inferior y presente la pendiente descendente menos pronunciada. Este enfoque asegura que todas las aguas servidas fluyan naturalmente hacia el nodo sumidero. En el algoritmo 5 se presenta el pseudocódigo de esta primera etapa y en la figura 4.4 se presenta el diseño final del grafo una vez finalizada la primera etapa, en la cual el nodo sumidero es representado por el punto de color rojo. El diseño del escenario

para este ejemplo es que le se presenta en la figura 4.5. Para la construcción de la topología, se adoptó una heurística que conecta cada nodo con su vecino de cota inferior que presenta la pendiente más suave. La hipótesis detrás de esta elección es que, al minimizar el desnivel en cada paso, se podría limitar la necesidad de excavaciones profundas aguas abajo. Si bien esta decisión no garantiza el óptimo global, ya que un camino localmente plano podría llevar a una ruta global más larga y costosa, su principal fortaleza es la simplicidad y rapidez. Es importante destacar que la progresión de este algoritmo está garantizada por la naturaleza de los datos de entrada. El escenario hipotético en el que un nodo esté rodeado exclusivamente por vecinos a una altura exactamente igual es, en la práctica, inviable. Esto se debe a que las alturas de los nodos provienen de una superficie de elevación continua generada mediante una interpolación TIN. Dicho proceso, al operar con valores de punto flotante, asegura la existencia de diferenciales de cota, aunque sean infinitesimales, entre coordenadas geográficas distintas, garantizando que siempre exista un camino descendente para cada nodo en las instancias estudiadas.

Algoritmo 5 Primera etapa del algoritmo greedy

Entrada: Grafo del escenario G_{base}, Nodo sumidero N_{pozo}
Salida: Grafo dirigido conexo $G = (V, E)$ con la topología de la red

G_{solucion} = GrafoDirigido con todos los nodos de G_{base}
nodos_a_conectar = nodos en G_{base} excepto N_{pozo}
for nodo u en nodos_a_conectar **do**
    altura_u = obtener_altura(u)
    candidatos = []
    for cada vecino v de u en G_{base} **do**
        altura_v = obtener_altura(v)
        if $\text{altura}_v < \text{altura}_u$ **then**
            distancia = obtener_distancia(u, v)
            pendiente = $(\text{altura}_u - \text{altura}_v)$/distancia
            agregar {vecino: v, pendiente: pendiente} a candidatos
        end if
    end for
    if candidatos no está vacío **then**
        ordenar candidatos por pendiente (ascendente)
        vecino_elegido = seleccionar_mejor_candidato(candidatos)
        agregar_arista(u, vecino_elegido) en G_{solucion}
    end if
end for
return G_{solucion}

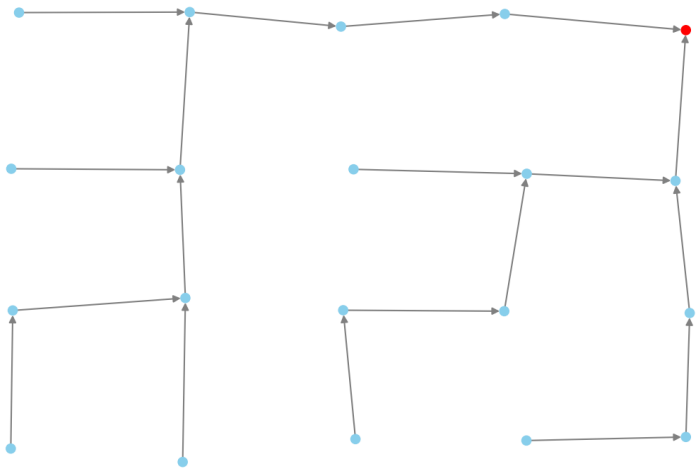


Figura 4.4: Diseño de la red tras la primera etapa

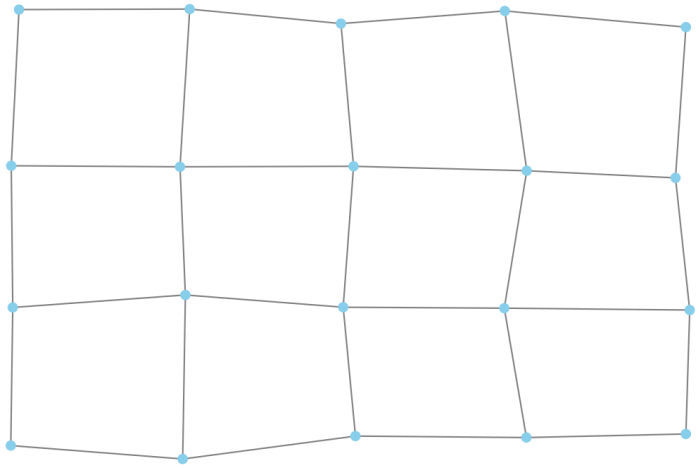


Figura 4.5: Diseño del escenario

Una vez definida la topología una segunda fase recorre la red para asignar los atributos geométricos e hidráulicos (diámetro, pendiente real y profundidad

del pozo) a cada tramo, asegurando la factibilidad de la solución y optimizando los costos localmente. Un ordenamiento topológico es una forma de ordenar los nodos de un grafo dirigido y acíclico de manera que para cada arista (u, v), u siempre aparece antes que v en la secuencia. En la segunda etapa del algoritmo greedy se itera sobre los nodos en orden topológico. Esta decisión garantiza que al momento de procesar una tubería, ya se ha calculado y acumulado el caudal total proveniente de todas las conexiones aguas arriba, permitiendo tomar la decisión óptima local con la demanda real que deberá soportar cada tubería. La secuencia de la segunda etapa, la cual se encuentra presentada en el algoritmo 6, es la siguiente. Para cada tramo (u, v) el algoritmo realiza:

- Cálculo del caudal: Determina la demanda total que deberá soportar la tubería, sumando la demanda local del tramo con la demanda ya acumulada en el nodo aguas arriba u.
- Búsqueda de conexión factible: Busca la combinación de diámetro y pendiente que sea hidráulicamente factible y respete las restricciones geográficas para el caudal calculado.
- Asignación de atributos: Almacena en la arista (u, v) el diámetro y la pendiente encontrados.
- Actualización geométrica: En caso de ser necesario, actualiza la profundidad del pozo del nodo aguas abajo v para asegurar la correcta conexión física de la tubería.

La función *mejor_conexion* del algoritmo 6 se utiliza para encontrar la combinación óptima de pendiente y diámetro para la conexión (u, v). El pseudocodigo de esta función se encuentra presentado en el algoritmo 7. El algoritmo para obtener la solución óptima, para cada tubería (u, v) controla que

- Cumpla con todas las restricciones hidráulicas
- Cumpla con las restricciones geométricas (profundidad mínima de instalación)
- Tenga el menor costo de construcción entre todas las opciones

Cada combinación de pendiente y diámetro que resulte en una conexión hidráulicamente factible es sometida a un proceso de verificación geométrica para asegurar que la profundidad de la tubería respeta las medidas mínimas establecidas. Si la solución es válida geométricamente, se calcula su costo de

construcción, el cual se compone del costo de la tubería como el de la excavación necesaria. El algoritmo almacena la solución factible de menor costo encontrada hasta el momento. El proceso se repite para todas las pendientes válidas, garantizando que la solución final seleccionada para el tramo sea la más económica posible. La función *es_factible_hidraulicamente* se presentará más adelante en la sección 4.2.2.1, en el algoritmo 8. Una vez encontrada una conexión factible para el tramo (u, v) con una pendiente s_i, se calcula la altura a la cual se realizará la conexión con el pozo destino v. Esta altura es la cota a la que llegará la nueva tubería y se obtiene de la siguiente forma $altura\_conexion_v = altura\_pozo_u - s_i * distancia(u, v)$. Para garantizar una red físicamente conectada, la altura del pozo del nodo v debe ser como mínimo igual a la altura de la conexión entrante más baja. Por lo tanto, el algoritmo actualiza dinámicamente la altura del pozo v al mínimo valor entre su altura actual y la nueva altura de conexión, como se ilustra en la figura 4.6. De esta forma se asegura que a medida que la red se diseña desde aguas arriba hacia aguas abajo la profundidad de los pozos se ajusta para recibir correctamente todos los flujos entrantes.

Algoritmo 6 Generación de red hidráulicamente factible

Entrada: Grafo dirigido G_{solucion} con la topología de la red
Salida: G_{solucion} factible hidráulicamente

```
1:  nodos_ordenados = orden_topologico(G_solucion)
2:  for cada nodo u en nodos_ordenados do
3:      if u es el nodo sumidero global then
4:          continue
5:      end if
6:      v = sucesor de u en G_solucion
7:      demanda = demanda_propia(u, v) + demanda_acumulada(u)
8:      actualizar_demanda_acumulada(v, demanda)
9:      distancia = obtener_distancia(u, v)
10:     conexion_optima = mejor_conexion(u, v, demanda, distancia, G_solucion)
11:     if existe conexion_optima then
12:         asignar_atributos_a_arista(u, v, conexion_optima)
13:         actualizar_altura_pozo(v, conexion_optima.altura_conexion)
14:     end if
15: end for
16: return G_solucion
```

A partir de la solución construida en esta sección, es posible recuperar los datos necesarios para calcular el costo total de construcción de la red,

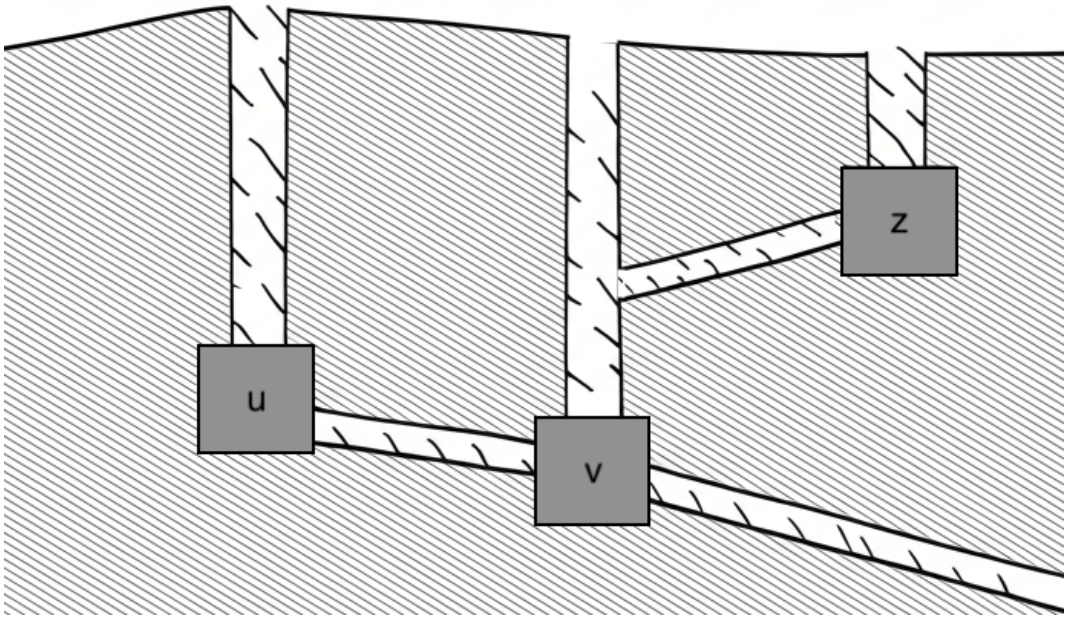


Figura 4.6: Representación del ajuste de la altura de un pozo en función de las alturas de conexión. El pozo 'v' recibe conexiones de los pozos 'u' y 'z' a diferentes elevaciones. La profundidad final del pozo 'v' se establece para poder recibir la conexión más baja y así garantizar el flujo gravitacional de toda la red que drena hacia él

utilizando los atributos presentes en el grafo. Entre estos datos se incluye la altura a la cual se encuentra el pozo, que se almacena en los vértices, así como el largo de las tuberías, altura de conexión y el caudal, presentes en las aristas.

En resumen, en esta sección se describió un algoritmo greedy de dos fases para el diseño de redes de saneamiento. La primera fase estableció una topología de red conectada basada en una heurística de pendiente suave, mientras que la segunda fase asignó atributos factibles y de bajo costo recorriendo la red en orden topológico. Si bien la solución obtenida con este método puede no ser la óptima global, proporciona una solución de alta calidad y sirve como una base sólida para la comparación y la inicialización de la población del algoritmo evolutivo que se presenta a continuación.

Algoritmo 7 mejor_conexión

Entrada: Nodos u, v, Caudal Q, Distancia, Grafo actual G_{actual}
Salida: La mejor conexión encontrada (diccionario con atributos) o Nulo

```
solucion = Nulo
costo_minimo = infinito
altura_pozo_u = obtener_altura_pozo(u, G_actual)
for cada pendiente S en el rango de pendientes permitidas do
    for cada diámetro D en la lista de diámetros disponibles do
        if es_factible_hidraulicamente(Q, S, D) then
            alt_conexion = altura_pozo_u − (S × Distancia)
            if cumple_restriccion_profundidad(altura_conexion, v) then
                costo_actual = calcular_costo_tramo(D, S, u, v)
                if costo_actual < costo_minimo then
                    costo_minimo = costo_actual
                    solucion = {diámetro:D, pendiente:S, altura_conexion: alt_conexion}
                end if
                break
            end if
        end if
    end for
end for
return solucion
```

4.2.2. Algoritmo evolutivo

La aplicación de algoritmos evolutivos en los problemas de optimización en el trazado de redes de saneamiento ofrece una solución eficaz para abordar los desafíos complejos que enfrenta este tipo de sistemas. Para la implementación de este algoritmo, se utilizó el framework de computación evolutiva DEAP (Fortin et al. 2012), una biblioteca de Python que provee las estructuras de datos y los operadores necesarios para construir algoritmos evolutivos. Estos algoritmos, inspirados en los principios de la evolución natural, permiten explorar grandes espacios de búsqueda en busca de soluciones de alta calidad, incluso cuando la función objetivo no es lineal ni diferenciable, lo que los hace especialmente adecuados para problemas donde existen múltiples restricciones técnicas, geográficas y económicas. Diversos trabajos en la literatura han demostrado su efectividad en este contexto. Por ejemplo, Hassan et al. (2020) aplican un algoritmo genético híbrido al diseño óptimo de redes de alcantarillado, obteniendo resultados de alta calidad y superando los métodos tradicionales

en varios casos. Este tipo de enfoques resulta particularmente valioso cuando la búsqueda exhaustiva se vuelve impracticable.

En esta sección, se explora cómo los algoritmos evolutivos pueden modelar y mejorar la eficiencia de la construcción de redes de saneamiento, en comparación con un algoritmo greedy, imitando el proceso de selección natural para encontrar soluciones óptimas o cercanas a óptimas.

Realizar una búsqueda por fuerza bruta de una solución no es una forma inteligente de resolver el problema, debido al gran tamaño del espacio de soluciones. Esta complejidad surge de la gran cantidad de variables que deben considerarse, tanto para cumplir con las restricciones impuestas por el terreno como con las condiciones hidráulicas del sistema. Es aquí donde los algoritmos evolutivos se desempeñan de manera eficaz, gracias a sus mecanismos para explorar el espacio de soluciones de forma eficiente.

La descripción del algoritmo evolutivo se estructura de la siguiente manera: primero se define la función objetivo y las restricciones. Luego, se presenta la representación de las soluciones. Se continúa con el detalle de los operadores genéticos utilizados. Finalmente, se explica el proceso de corrección de soluciones no factibles.

4.2.2.1. Función objetivo

En esta sección, se define la función objetivo que se utiliza para evaluar la calidad de las soluciones obtenidas mediante el algoritmo evolutivo. Como se mencionó anteriormente, el problema a resolver consta de minimizar el costo de construcción de una red de saneamiento para una ciudad, o sección de una ciudad atada a restricciones geográficas e hidráulicas. Según los trabajos relevados y como se discute en el capítulo 3, existe un consenso entre los autores en que las principales características que influyen en el costo total de la red de saneamiento son el costo de la mano de obra en la región, la cantidad y calidad de los materiales utilizados y la resistencia del terreno. A su vez, tanto las regulaciones existentes como las restricciones geográficas y/o hidráulicas repercuten en el diseño final de la red y, por lo tanto, en el costo total de la obra.

En la ecuación 4.1 se detalla la función objetivo a minimizar por el algoritmo evolutivo. En donde CS_i representa el costo de construcción de la tubería en el tramo i y CM_i representa el costo de construcción del pozo de registro.

El costo CS_i incluye el costo de la tubería según su largo L y su diámetro D así como también el costo de la excavación para su colocación.

$$\text{mín}\, C = \sum_{i=1}^{N} CS_i + CM_i \tag{4.1}$$

Entre las restricciones a las cuales se encuentra atada la solución se encuentran la pendiente utilizada, las profundidades de excavación, los diámetros de tuberías disponibles a utilizar en cada tramo, la lámina máxima y la fuerza de arrastre mínima. Estas restricciones permiten que la solución hallada cumpla con las regulaciones existentes, así como también aseguran el correcto funcionamiento hidráulico de la red.

1. $S_{minima} \leq S_i \leq S_{maxima}$
   Donde S_i hace referencia a la pendiente de la tubería i
2. $P_{minima} \leq P_i$
   Donde P_i hace referencia a la profundidad del pozo i.
3. $D_i \in D$
   Siendo D el conjunto con todos los diámetros de tuberías disponibles
4. $D_i \leq D_{i+1}$
   Referencia que el diámetro de la tubería D_i debe ser menor o igual que el diámetro de la tubería D_{i+1} aguas abajo.
5. $\beta_i \leq \beta_{max}$
   Referencia a la lámina o altura del agua que fluye a través de la tubería i, esta se calcula de la forma $\beta_i = \frac{y_i}{D_i}$, donde y_i es la profundidad del flujo y D_i es el diámetro de la tubería utilizada en el tramo i.
6. $F_{min} \leq F_i$
   Donde F_i referencia a la fuerza de arrastre de las aguas servidas en la tubería i

Para verificar el cumplimiento de las restricciones hidráulicas de lámina máxima y fuerza de arrastre mínima, es necesario realizar un análisis detallado del flujo para cada tramo de tubería i de forma tal de hallar los valores y_i y F_i que permitan calcular las restricciones.

Para una demanda Q_i (correspondiente a la demanda acumulada en el tramo i), un diámetro D_i, una pendiente S_i y un coeficiente de rugosidad de Manning n (en este trabajo se utilizó $n = 0{,}01$), no es posible despejar de

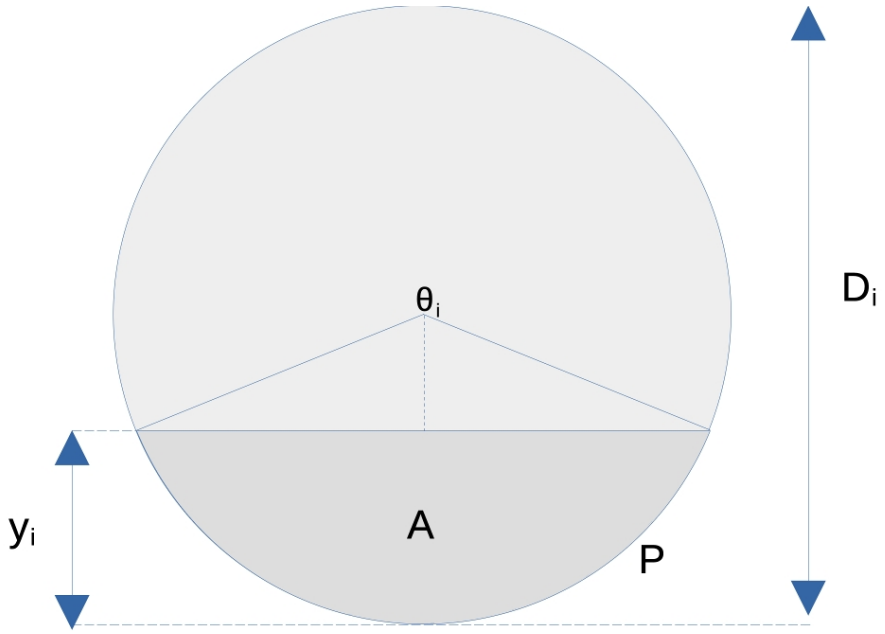


Figura 4.7: Sección transversal de la tubería i operando parcialmente llena. Se muestra el Diámetro (D_i), el tirante de agua (y_i), el Área Mojada (A), el Perímetro Mojado (P) y el ángulo central (θ) subtendido por la superficie libre.

forma analítica la profundidad del flujo y_i ni el ángulo central θ a partir de la ecuación de Manning (ecuación 4.2).

En dicha ecuación, el área mojada A y el radio hidráulico $R_h = A/P$ dependen de manera no lineal e implícita de la geometría de la sección, determinada por D_i y y_i (o θ). Por esta razón, se recurre a un método numérico iterativo para resolver la ecuación y obtener el valor de y_i (o su equivalente angular θ) que satisface el caudal Q_i. Este procedimiento se implementa dentro de la función de evaluación de factibilidad hidráulica, presentada en el algoritmo 8.

$$Q_i = \frac{1}{n} A \left(R_h^{2/3} S_i^{1/2} \right) \tag{4.2}$$

Una vez calculado el ángulo central θ mediante el procedimiento iterativo, se determinan explícitamente las propiedades geométricas e hidráulicas:

- Área mojada: $A = \frac{D_i^2}{8}(\theta - \sin(\theta))$
- Perímetro mojado: $P = \frac{D_i}{2}\theta$
- Radio hidráulico: $R_h = \frac{A}{P} = \frac{D_i}{4}\left(1 - \frac{\sin(\theta)}{\theta}\right)$
- Profundidad del flujo: $y_i = \frac{D_i}{2}\left(1 - \cos\left(\frac{\theta}{2}\right)\right)$

A partir de estos valores, se verifican las restricciones hidráulicas correspondientes al tramo i:

- Lámina relativa (restricción 5): $\beta_i = \frac{y_i}{D_i}$
- Fuerza de arrastre (restricción 6): $F_i = \gamma \cdot R_h \cdot S_i$

En estas expresiones, γ representa el peso específico del fluido. En el caso de aguas servidas, se utiliza una aproximación al valor correspondiente al agua: $\gamma \approx 9810\, N/m^3$.

Este procedimiento de cálculo se aplica a cada tramo de tubería (i) perteneciente a una solución candidata generada por el algoritmo evolutivo. Una solución se clasifica como factible únicamente si todos sus tramos cumplen con las restricciones hidráulicas y de diseño enumeradas (1 a 6). En caso contrario, se aplica el proceso de corrección definido en la sección 4.2.2.7.

Algoritmo 8 Factibilidad hidráulica

Entrada: caudal, pendiente y diámetro
Salida: booleano indicando si la tubería con la pendiente y diámetro indicado es factible según el caudal

coeficiente_manning = 0,01
lamina_maxima = 0,75
fuerza_arrastre_min = 1,0
$E = \dfrac{\text{coeficiente\_manning} \times (\text{caudal}/1000)}{\sqrt{\text{pendiente}} \times (\text{diámetro}/1000)^{8/3}}$
$E_2 = 8 \times \left(\dfrac{E^3}{4}\right)^{0,2}$
$K = E_2 \times \pi^{0,4} + \sin \pi$
for $i = 1$ **to** 13 **do**
    $K = E_2 \times K^{0,4} + \sin K$
end for
$T = E_2 \times K^{0,4} + \sin K$
if $T > 2\pi$ **then**
    $Y = 0$
    radio_hidraulico = 0
else
    $Y = \dfrac{(1 - \cos \frac{T}{2}) \times \text{diámetro}}{2000}$
    $\text{radio\_hidraulico} = \dfrac{\text{diámetro}}{4000} \times \dfrac{T - \sin T}{T}$
end if
$\text{lamina} = \dfrac{Y}{\text{diámetro}/1000}$
tension = 10000 × radio_hidraulico × pendiente
return lamina ≤ lamina_maxima **and** fuerza_arrastre_min ≤ tension

4.2.2.2. Representación

La forma en que se modelan las posibles soluciones es fundamental para que puedan ser procesadas por un AE. Esta sección detalla el esquema de representación adoptado para mapear soluciones al problema de construcción de redes de saneamiento dentro del AE implementado.

En el contexto de los AE, un cromosoma representa una solución candidata y se estructura como una cadena de datos que codifica sus componentes. Cada cromosoma está compuesto por genes, los cuales almacenan información específica sobre distintos aspectos de la solución. A su vez, los alelos son los valores concretos que toma cada gen.

El enfoque adoptado aborda simultáneamente el diseño de la red y la factibilidad hidráulica, por lo cual cada cromosoma incorpora la información necesaria para representar ambos aspectos de manera integrada. En esta representación, cada gen está asociado a un nodo de la red y contiene información sobre su conexión aguas abajo. Para ello, cada nodo se representa mediante un cuarteto de genes contiguos, estructurado de la siguiente forma:

1. El primer gen indica el identificador del nodo siguiente en la cadena de flujo.
2. El segundo gen almacena el valor de la pendiente asociada a dicha conexión.
3. El tercer gen codifica el diámetro de la tubería a utilizar.
4. El cuarto gen representa la profundidad del pozo en el nodo de origen.

Dado que cada nodo se representa mediante cuatro genes, la longitud total de un cromosoma es igual a $cantidad\_nodos \times 4$.

En la figura 4.8 se representa un ejemplo de una sección de un cromosoma con tres cuartetos de genes separados por colores con los alelos correspondientes según lo detallado anteriormente. Para identificar a cada nodo origen, se utiliza la posición en la que se encuentra el cuarteto que representa al nodo dentro de la cadena del cromosoma. De esta forma, el nodo con identificador 0 es representado por los primeros cuatro genes de la cadena, el nodo con identificador 1 es representado por los siguientes cuatro genes de la cadena y así sucesivamente.

Aunque la conexión de salida del nodo sumidero global es fija (representada por un gen con valor -1), es fundamental incluirlo en el cromosoma. Esto se

debe a que su cuarto gen, la profundidad del pozo, es una variable dependiente del resto de la red. Su valor final, determinado por las cotas de las tuberías que llegan a él, es crucial para calcular el costo total de excavación, que es un componente esencial de la función objetivo.

4.2.2.3. Inicialización

El proceso de inicialización establece el punto de partida para la búsqueda de soluciones óptimas por parte del AE. En esta sección se detalla cómo se genera la población inicial de soluciones candidatas, con un fuerte énfasis en la diversidad. Para la generación de la población inicial, se utiliza una versión modificada de el algoritmo greedy descrito en la sección 4.2.1, de forma de agregar diversidad en las soluciones generadas. Un algoritmo greedy, por definición, elige siempre la solución localmente óptima en cada paso. Para introducir diversidad, en cada paso se elige aleatoriamente una de las mejores N opciones disponibles. Así, se controla cuán cercana a la solución provista por el algoritmo greedy se encuentra una solución modificando el valor de N. En el proceso de generación de la población inicial, se asegura que siempre exista un individuo que represente la solución óptima del algoritmo greedy, fijando el valor de N en 1. Los siguientes individuos se generan aumentando progresivamente el valor de N, hasta que en cada paso se contemplen todas las opciones posibles.

4.2.2.4. Selección

El operador evolutivo de selección en un AE determina qué individuos sobreviven y se reproducen en cada generación. La importancia de este mecanismo radica en que un buen método de selección permite preservar la diversidad genética y explorar el espacio de soluciones, además de explotar los mejores individuos en la búsqueda del óptimo.

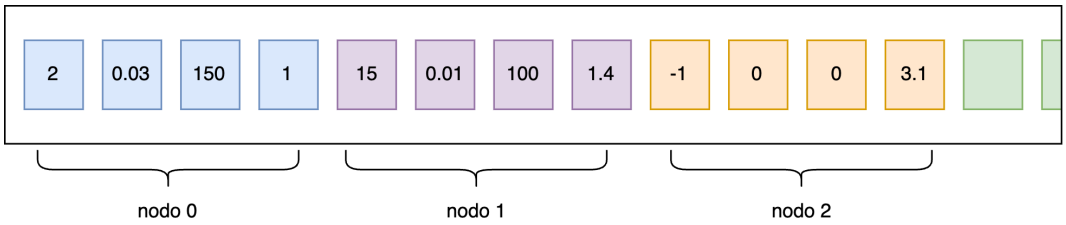


Figura 4.8: Estructura de un cromosoma

El elitismo en un AE garantiza que las mejores soluciones de cada generación se transfieran a la siguiente, evitando la pérdida de soluciones valiosas. Sin embargo, una presión elitista elevada puede limitar la exploración del espacio de soluciones. Para mitigar este efecto, se restringe el tamaño de la selección elitista a tres individuos.

En la solución propuesta, se implementó una selección que combina la selección por torneo y el elitismo. Como primer paso, se almacenan los mejores individuos de la generación, y luego se aplica selección por torneo sobre los individuos restantes. Esta combinación mantiene un equilibrio entre preservar soluciones de alta calidad y explorar nuevas áreas del espacio de soluciones.

4.2.2.5. Cruzamiento

El cruzamiento es el proceso mediante el cual se crean nuevos individuos, denominados hijos, a partir de la combinación de características de otros individuos, denominados padres. El proceso de cruzamiento comienza eligiendo dos individuos y combinando ciertas características de estos para producir dos nuevos individuos, los cuales remplazarán a sus padres en la siguiente generación. Con el cruzamiento se busca explotar la posibilidad de que estos nuevos individuos hereden las mejores características de sus padres dando lugar a mejores soluciones. En esta sección se explica el cruzamiento implementado.

Por simplicidad, al momento de realizar el cruzamiento se decodificaron los individuos a sus respectivos fenotipos. De esta forma, en lugar de trabajar directamente con la representación de la solución, pasamos a trabajar con grafos. Dado que el grafo es la representación completa de una solución, es en los nodos y aristas del grafo que se encuentran los candidatos a cruzar para generar los nuevos individuos. A grandes rasgos, el cruzamiento implementado utiliza subgrafos generados a partir de cada individuo padre para generar los hijos.

En el algoritmo 9 se describe el pseudocódigo del cruzamiento implementado. El proceso para generar cada uno de los hijos es similar, se modifica únicamente el padre inicial sobre el cual se trabaja. Como primer paso se divide el grafo *padre_1* en dos grafos distintos. Esta división se realiza eliminando una arista del grafo, con la condición de que esta no sea incidente al nodo pozo global ni parta desde un nodo con grado de entrada 0. Luego de esto, se elige entre los subgrafos generados cual será el que se utilice como representante del

padre_1. En la solución desarrollada, esta selección fue implementada de manera aleatoria. Al nuevo grafo obtenido se le asigna el nombre de *subgrafo_1*, el cual representa a la característica del grafo *padre_1* a utilizar en la generación del *hijo_1*. La característica proveniente del *padre_2* a propagar a la siguiente generación se consigue generando un nuevo grafo de nombre *padre_2_aux* del cual se eliminan los nodos y sus respectivas aristas adyacentes que también se encuentren en el grafo *subgrafo_1*. Por último, el *hijo_1* es generado como la unión de los grafos *subgrafo_1* y *padre_2_aux*. Para generar al grafo *hijo_2* se realiza un procedimiento similar, modificando la primera selección de *padre_1* por *padre_2*.

Algoritmo 9 Cruzamiento

Entrada: Grafos padre_1 y padre_2
Salida: Grafos hijo_1 e hijo_2 a incluirse en la próxima generación

```
1: corte_1 = generar_subgrafo(padre_1)
2: subgrafo_1 = elegir_subgrafo_utilizar(corte_1)
3: padre_2_aux.eliminar_nodos(subgrafo_1, padre_2)
4: hijo_1 = padre_2_aux.union(subgrafo_1)
5: corte_2 = generar_subgrafo(padre_2)
6: subgrafo_2 = elegir_subgrafo_utilizar(corte_2)
7: padre_1_aux.eliminar_nodos(subgrafo_2, padre_1)
8: hijo_2 = padre_1_aux.union(subgrafo_2)
```

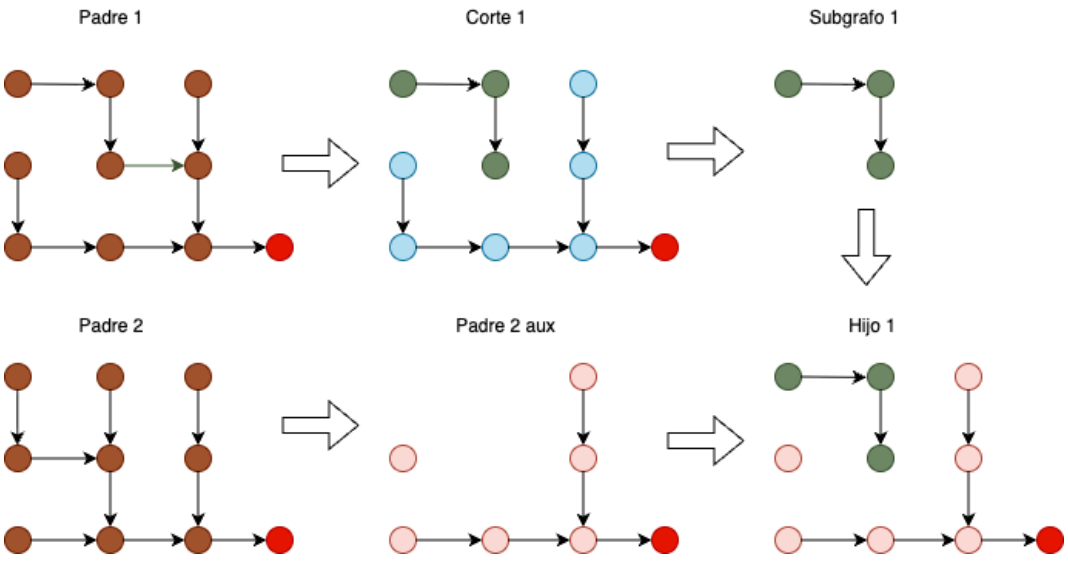


Figura 4.9: Representación del cruzamiento para el hijo 1

En la figura 4.9 se ilustra el proceso de cruzamiento explicado previamente. Las representaciones de los descendientes obtenidos por el operador del cruzamiento son grafos disconexos, ya que fueron generados a partir de la unión de subgrafos distintos. Por lo tanto, las soluciones que representan no son factibles en esta etapa. La sección 4.2.2.7 describe el mecanismo mediante el cual se incorporan los enlaces faltantes para convertir estos individuos en soluciones viables.

4.2.2.6. Mutación

La mutación es un operador evolutivo de exploración cuyo objetivo es introducir cambios aleatorios en la información genética de los individuos. Al realizar estas modificaciones, se exploran nuevas áreas del espacio de soluciones, lo que brinda la posibilidad de escapar de óptimos locales y evita la convergencia prematura a soluciones subóptimas. Por lo tanto, este es un operador crucial para mantener la diversidad genética de los individuos, lo que a su vez favorece a una búsqueda efectiva de soluciones. Al ser un operador cuya aplicación produce grandes cambios en los individuos de la población, se le asigna una probabilidad baja de aplicación, de forma de no perder las buenas soluciones obtenidas.

En la solución desarrollada se trabajó con dos probabilidades distintas para la implementación de la mutación. La primera de ellas, denominada p, es utilizada en cada generación para sortear cuáles individuos mutar y cuales no. De modo de no perder buenas soluciones en generaciones tardías, es que se usa una probabilidad p baja. La segunda probabilidad utilizada, denominada $p\_interna$, es utilizada sobre cada individuo seleccionado para mutar e indica cuáles genes de este serán mutados y cuales no.

En el algoritmo 10 se presenta el pseudocódigo del operador de mutación implementado. Este operador permite modificar, de forma aislada o combinada, los diámetros, las pendientes y los enlaces existentes entre nodos. En cada generación, y dada una probabilidad p, se seleccionan los individuos a mutar. Luego, para cada uno de ellos, se determinan aleatoriamente las características que serán modificadas. Este procedimiento se ve en la línea 2 del algoritmo 10, donde se selecciona al menos una característica del individuo de forma aleatoria.

Algoritmo 10 Mutación

Entrada: individuo **ind** a mutar y probabilidad **p_interna**
Salida: individuo **ind** mutado

```
1: mutaciones = ["mutar_pendiente", "mutar_diametro", "mutar_enlace"]
2: mutaciones_a_aplicar = obtener_mutaciones_a_aplicar(mutaciones)
3: for mutacion in mutaciones_a_aplicar do
4:     if mutacion = "mutar_pendiente" then
5:         ind = mutar_pendiente(ind, p_interna)
6:     end if
7:     if mutacion = "mutar_diametro" then
8:         ind = mutar_diametro(ind, p_interna)
9:     end if
10:    if mutacion = "mutar_enlace" then
11:        ind = mutar_enlace(ind, p_interna)
12:    end if
13: end for
```

En los algoritmos 11, 12 y 13 se describe el pseudocódigo de las implementaciones utilizadas para mutar las pendientes, los diámetros y los enlaces de los individuos. Como parámetros de entrada se tienen al individuo a mutar y a la probabilidad *p_interna*. En cada uno de estos algoritmos, se recorren todos los genes que componen al individuo y dada la probabilidad *p_interna* se decide si mutar la característica correspondiente al algoritmo o no. La modificación de cada gen se hace en el alelo correspondiente a la característica según lo visto en la sección 4.2.2.2.

Algoritmo 11 Mutar pendiente

Entrada: individuo **ind** a mutar y probabilidad **p_interna**
Salida: individuo **ind** mutado

```
1: cantidad_genes = length(ind)
2: i = 0
3: while i+4 ≤ cantidad_genes do
4:     p_interna_aux = random(0,1) //Obtiene un número al azar entre 0 y 1
5:     if p_interna_aux ≤ p_interna then
6:         nueva_pendiente = obtener_nueva_pendiente()
7:         ind[i+1] = nueva_pendiente
8:     end if
9:     i = i + 4
10: end while
```

El algoritmo 13 no conecta el nodo (representado por el gen mutado) con otro, sino que simplemente elimina el enlace. Durante la fase de corrección

Algoritmo 12 Mutar diámetro

Entrada: individuo **ind** a mutar y probabilidad **p_interna**
Salida: individuo **ind** mutado

```
1: cantidad_genes = length(ind)
2: i = 0
3: while i+4 ≤ cantidad_genes do
4:     p_interna_aux = random(0,1) //Obtiene un número al azar entre 0 y 1
5:     if p_interna_aux ≤ p_interna then
6:         nuevo_diametro = obtener_nuevo_diametro()
7:         ind[i+2] = nuevo_diametro
8:     end if
9:     i = i + 4
10: end while
```

Algoritmo 13 Mutar enlace

Entrada: individuo **ind** a mutar y probabilidad **p_interna**
Salida: individuo **ind** mutado

```
1: cantidad_genes = length(ind)
2: i = 0
3: while i+4 ≤ cantidad_genes do
4:     p_interna_aux = random(0,1) //Obtiene un número al azar entre 0 y 1
5:     if p_interna_aux ≤ p_interna then
6:         ind[i] = -1
7:     end if
8:     i = i + 4
9: end while
```

de los individuos, detallada en la sección 4.2.2.7, se describe el proceso para agregar los enlaces faltantes de modo tal que el individuo quede nuevamente factible.

En la figura 4.10 se presentan dos ejemplos de individuos mutados. En el individuo 1 se mutan la pendiente y el diámetro del segundo y tercer gen. Mientras que del individuo 2 se muta únicamente el enlace del primer y tercer gen del cromosoma.

4.2.2.7. Corrección

Esta sección describe el procedimiento aplicado para corregir individuos no factibles generados durante la ejecución de los operadores evolutivos, con el fin de convertirlos en soluciones viables que cumplan con todas las restricciones impuestas por el problema.

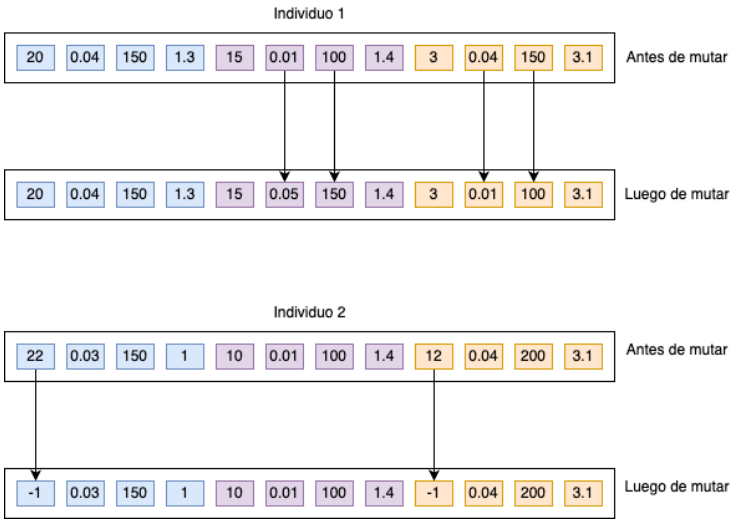


Figura 4.10: Mutación

Durante el proceso evolutivo, puede ocurrir que algunos de los nuevos individuos generados no satisfagan las restricciones del problema. Existen diversas estrategias para abordar esta situación, entre ellas: rechazar al individuo y generar uno nuevo, aplicar una penalización a su valor de fitness, o bien modificarlo para que se torne factible.

La selección de una estrategia adecuada depende de las características específicas del problema considerado. En este trabajo, la generación de un nuevo individuo conlleva un costo computacional elevado, por lo cual la opción de descartarlo no resulta eficiente. Por otra parte, como se expone en las secciones 4.2.2.5 y 4.2.2.6, los individuos generados mediante cruzamiento o mutación de enlaces no representan grafos conexos, lo cual los invalida como soluciones viables.

Dado este contexto, se adopta la estrategia de corrección, que permite garantizar la factibilidad de las soluciones antes de ser evaluadas por la función objetivo.

En el algoritmo 14 se presenta el procedimiento general implementado para corregir individuos no factibles. El proceso de reparación es un procedimiento iterativo, que se repite hasta que la solución candidata alcanza un estado estable y completamente factible. Este enfoque es necesario debido a que la corrección de un aspecto de la red puede afectar la factibilidad de otro. En cada iteración del ciclo de reparación se aplican secuencialmente los siguientes subprocesos de corrección:

Algoritmo 14 Proceso iterativo de corrección de soluciones

Entrada: Grafo inicial potencialmente no factible G_{inicial}
Salida: Grafo final factible y estable G_{final}, o Nulo si la reparación falla

$G_{\text{actual}} = G_{\text{inicial}}$.copy()
for $i = 1$ to max_iteraciones **do**
    $G_{\text{anterior}} = G_{\text{actual}}$
    $G_{\text{temp}} = \text{sincronizar\_geometria}(G_{\text{anterior}})$
    $G_{\text{temp}} = \text{agregar\_enlaces\_faltantes}(G_{\text{temp}})$
    completar_demanda_grafo(G_{temp})
    $G_{\text{actual}} = \text{generar\_grafo\_factible}(G_{\text{temp}}, \text{sobrescribir\_opevol} = \text{False})$
    if G_{actual} es Nulo **then**
        $G_{\text{actual}} = \text{generar\_grafo\_factible}(G_{\text{temp}}, \text{sobrescribir\_opevol} = \text{True})$
    end if
    if G_{actual} es Nulo **then**
        return Nulo
    end if
    if son_grafos_identicos($G_{\text{anterior}}, G_{\text{actual}}$) **then**
        break
    end if
end for
return G_{actual}

1. Sincronización geométrica: Primero se realiza una validación de la geometría actual del grafo. Se eliminan las aristas que son físicamente inviables, como aquellas con pendiente nula o negativa. Este paso asegura una base geométrica coherente antes de continuar.
2. Reparación de conectividad: Luego, se verifica si el grafo es conexo. Si una operación previa (cruzamiento, mutación o sincronización geométrica) ha desconectado la red, se invoca a la función *agregar_enlaces_faltantes*, definida en el algoritmo 15, la cual se encarga de restablecer la conectividad.
3. Actualización de la demanda: Con la topología ya conectada, se recalcula el caudal en toda la red. Este proceso se realiza utilizando el ordenamiento topológico del grafo y completando la demanda desde los nodos aguas arriba hasta el sumidero, proceso similar al implementando en el algoritmo 6.
4. Reparación de factibilidad hidráulica: Finalmente se aplica la función *generar_grafo_factible*, que recorre la red para identificar y corregir cualquier tramo de tubería que no cumpla con las restricciones hidráulicas.

Esta ejecución se realiza en dos etapas. La primera se configura para que no sobrescriba los cambios introducidos por la mutación y el cruzamiento con el objetivo de mantener la diversidad genética introducida. Si con esas restricciones no fuera posible corregir al individuo se permite una modificación total del individuo.

5. Por último, si luego de ejecutar todas las iteraciones no fuera posible corregir al individuo, se descarta y se remplaza por uno nuevo aplicando el proceso de inicialización para un solo individuo.

La reparación de la conectividad (paso 2) es un paso crucial y se realiza mediante la función *agregar_enlaces_faltantes* cuyo pseudocódigo se presenta en el algoritmo 15. Este algoritmo identifica todas las componentes conexas y las une iterativamente hasta que solo queda una. En cada paso, se identifica la componente más cercana a la componente que contiene el nodo sumidero y se invoca a la función *unir_componentes* para crear un camino de bajo costo entre ellas. Para encontrar las conexiones, la función *unir_componentes*, se apoya en los nodos de menor altura, a los cuales llamamos sumideros locales, de cada subgrafo ya que estos son los que reciben todas las aguas servidas de su subgrafo. Si no existe una conexión directa, un método recursivo explora a los vecinos de los nodos sumideros locales para encontrar una ruta viable, modificando la red para preservar la factibilidad hidráulica.

Algoritmo 15 Agregar enlaces faltantes

Entrada: Grafo $G = (V, E)$
Salida: Grafo $G = (V, E)$ conexo, representando la red de saneamiento

```
1: grafos = obtener_componentes_conexos(G)
2: while len(grafos) > 1 do
3:     grafo_cercano = obtener_grafo_cercano_sumidero_global(grafos)
4:     G = unir_componentes(G, grafo_cercano, grafo_sumidero)
5:     grafos = obtener_componentes_conexos(G)
6: end while
```

En las figuras 4.11, 4.12, 4.13 y 4.14 se presenta un ejemplo de evolución del diseño de uno de los subgrafos del individuo en los pasos recursivos para encontrar una conexión con el nodo sumidero. En las figuras, el nodo sumidero se encuentra representado en color rojo y el nodo sumidero local del subgrafo de color negro.

Una vez asegurada la conectividad, el paso final del ciclo es la reparación hidráulica (paso 4). En el algoritmo 16 se presenta el pseudocódigo de la fun-

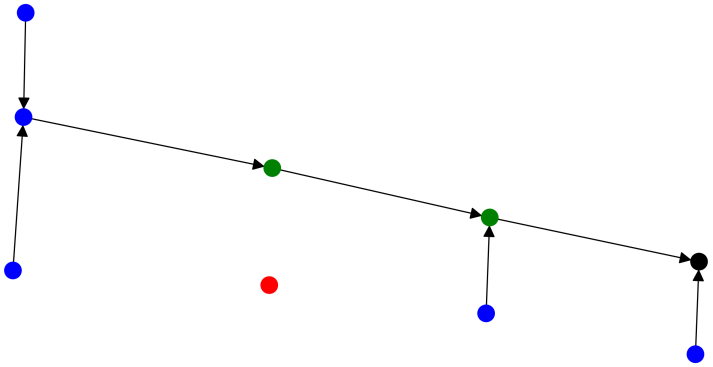


Figura 4.11: Subgrafo más cercano al nodo sumidero global en la red

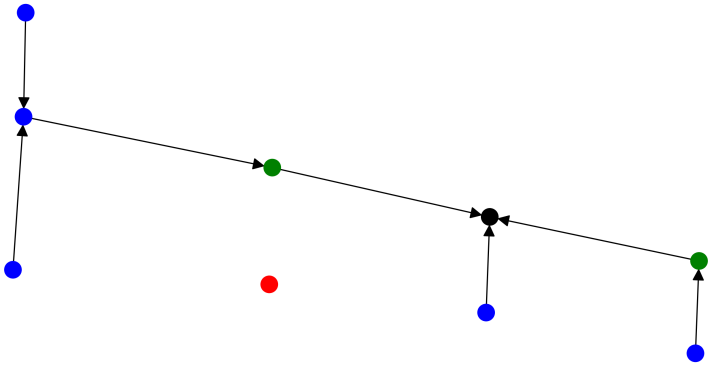


Figura 4.12: Primer paso en la recursión

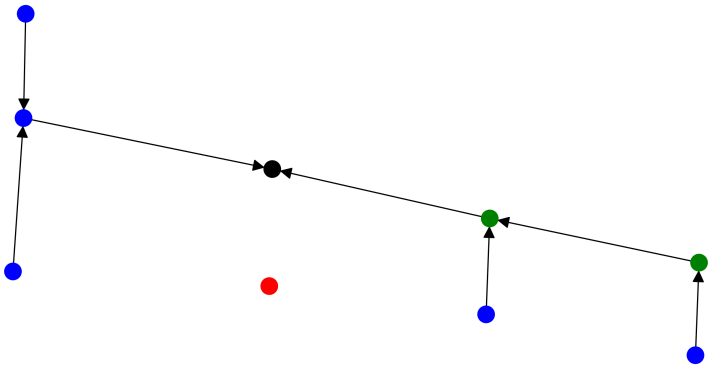


Figura 4.13: Segundo paso en la recursión

ción *generar_grafo_factible*, que es la responsable final de asegurar que todos los enlaces de la red con topología y demanda ya definidas cumpla con las

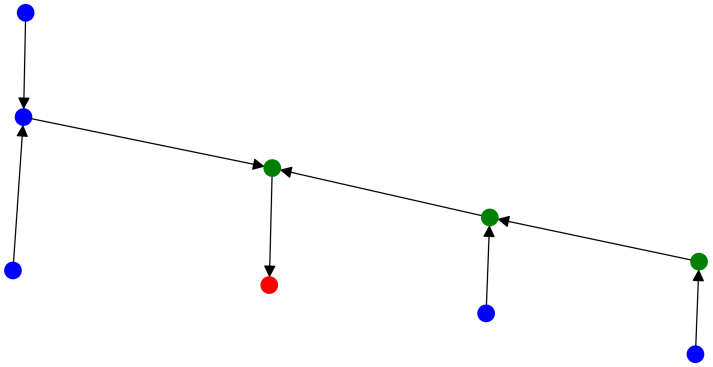


Figura 4.14: conexión entre el subgrafo y el nodo sumidero global

restricciones hidráulicas. Para ello, implementa una estrategia de verificación y reparación iterativa que recorre la red desde aguas abajo hacia aguas arriba, garantizando que todo el sistema sea coherente. El recorrido se gestiona mediante una estructura de pila (LIFO), que se inicializa con el nodo sumidero global. El algoritmo extrae un nodo de la pila y verifica la factibilidad de todas las tuberías que llegan a él desde sus nodos predecesores. Si un enlace es factible, su nodo predecesor se añade a la pila para ser analizado posteriormente, continuando de esta forma con el recorrido aguas arriba. Cuando se encuentra un enlace no factible, se invoca a la función de reparación *corregir_enlace* para modificar sus atributos (pendiente, diámetro) hasta generar una configuración válida. Si la reparación es exitosa, el proceso de validación se reinicia por completo vaciando la pila. El reinicio es fundamental, ya que la modificación de un enlace puede afectar la factibilidad de otros tramos ya validados como correctos.

La función *corregir_enlace* soluciona el problema de la siguiente forma:

1. Comienza un proceso iterativo donde prueba con todas las posibles pendientes
2. Para cada nueva altura de conexión, la función itera a través de todos los diámetros disponibles.
3. La primera combinación que cumpla con todas las restricciones hidráulicas se acepta como solución reparada para ese enlace.

Si el valor de *s_opevol* es False, y la mutación modificó la pendiente o el diámetro para este enlace, se corrige al individuo sin modificar el valor alterado. En

Algoritmo 16 Generar grafo factible

Entrada: Grafo no factible a corregir, **s_opevol**: condicional sobrescribir operadores evolutivos
Salida: grafo factible hidráulicamente o Nulo

```
1: grafo = agregar_enlaces_faltantes(grafo, opevol)
2: nodo_destino = obtener_nodo_sumidero_global(grafo)
3: pila = [(grafo, nodo_destino)]
4: while pila.size() > 0 do
5:     grafo, nodo_destino = pila.pop()
6:     if nodo_destino = null then
7:         nodo_destino = obtener_nodo_sumidero_global(grafo)
8:     end if
9:     predecesores = obtener_predecesores_nodo(grafo, nodo_destino)
10:    for origen in predecesores do
11:        if not cumple_factibilidad( grafo, origen, nodo_destino) then
12:            grafo = corregir_enlace( grafo, origen, nodo_destino, op_evol)
13:            nodo_destino = null
14:            pila.empty()
15:            pila = [(grafo, nodo_destino)]
16:            break
17:        end if
18:        if nodo_destino ≠ null then
19:            pila.add([grafo, origen])
20:        end if
21:    end for
22: end while
23: return grafo
```

caso de que *s_opevol* sea True, se permite modificar tanto la pendiente como el diámetro.

En este capítulo se detalló la metodología empleada en esta solución, comenzando por el proceso de armado y depuración del escenario, seguido de la descripción de los dos enfoques algorítmicos implementados: un algoritmo greedy y un algoritmo evolutivo. El siguiente capítulo se centrará en el análisis experimental y la evaluación del desempeño de las soluciones propuestas.

Capítulo 5

Análisis experimental

El objetivo de este capítulo consiste en la descripción detallada y el análisis de las pruebas que se realizaron durante el desarrollo del proyecto, explicando en cada caso los métodos y herramientas empleados.

5.1. Metodología de evaluación experimental

Este apartado describe el entorno en el cual se llevaron a cabo las pruebas, así como la estrategia adoptada para ajustar y evaluar los parámetros del algoritmo evolutivo propuesto. En primer lugar, se detalla el entorno de ejecución utilizado. Luego, se describen las metodologías estadísticas aplicadas para comparar configuraciones, y finalmente se presenta la configuración seleccionada para los experimentos finales.

5.1.1. Entorno de ejecución

Las pruebas computacionales de esta solución se llevaron a cabo en el Centro Nacional de Supercomputación (ClusterUY) (Nesmachnow y Iturriaga, 2019), una plataforma de alto desempeño diseñada para gestionar múltiples recursos de cómputo de manera integrada. Este sistema, ampliamente utilizado por científicos e investigadores en Uruguay, contó con una infraestructura de cómputo equivalente en potencia a más de 10.000 computadoras tradicionales, capaz de realizar complejas operaciones matemáticas y de procesamiento de datos en cuestión de segundos bajo el modelo de clúster. En particular, las simulaciones fueron ejecutadas en un nodo con el siguiente hardware: un

procesador Intel Xeon Gold 6138 de 40 núcleos, acompañado de 128 GB de memoria RAM.

5.1.2. Configuración paramétrica del algoritmo evolutivo

Debido a la naturaleza estocástica de los algoritmos evolutivos, los cuales pueden obtener diferentes resultados para distintas ejecuciones sobre la misma instancia del problema, se volvió imprescindible ajustar sus parámetros antes de llevar a cabo el análisis experimental. Con el objetivo de identificar la configuración que permitió obtener los mejores resultados, para cada configuración se realizaron 50 ejecuciones independientes del algoritmo, registrando en cada una la duración y el valor de fitness obtenido. Se recuerda que el objetivo del algoritmo es minimizar el costo total de la red diseñada.

Para la etapa de configuración paramétrica se definieron cuatro variables: la probabilidad de cruzamiento (p_c), la probabilidad de mutación (p_m), el número de generaciones (G) y el tamaño de la población (TP). Los valores considerados fueron $p_m \in \{0,01,\ 0,05,\ 0,1\}$, $p_c \in \{0,6,\ 0,75,\ 0,9\}$, $TP \in \{100,\ 200,\ 500\}$ y $G \in \{200,\ 500,\ 1000\}$. La probabilidad de mutación (p_m) se refiere a la probabilidad de que un individuo sea seleccionado para mutar en cada generación, como se describió en la sección 4.2.2.6. La segunda probabilidad, denominada *p_interna*, que determina si un gen específico dentro de un individuo seleccionado muta, se mantuvo con un valor fijo de 0.4 durante todos los experimentos. Se decidió fijar este valor para acotar el espacio de búsqueda de los parámetros y centrar el análisis en las variables de mayor impacto global.

Dado que el algoritmo tenía un tiempo de ejecución relativamente alto (alrededor de 30 minutos por corrida), se decidió no evaluar todas las combinaciones posibles, ya que esto implicaría ejecutar 81 configuraciones distintas. En su lugar, se optó por una estrategia en dos etapas que redujo la cantidad total a 18 combinaciones.

En una primera instancia, se mantuvieron constantes las probabilidades de cruzamiento y mutación (con $p_c = 0,75$ y $p_m = 0,05$) y se evaluaron las diferentes combinaciones entre tamaño de población y número de generaciones. La tabla 5.1 muestra las combinaciones analizadas. Los resultados de esta primera etapa indicaron que la configuración `POB_200_GEN_500` obtuvo los mejores desempeños, por lo que fue seleccionada para las pruebas posteriores. Se optó

por la configuración con un tamaño de población de 200 individuos en lugar de 500 debido al equilibrio entre la calidad de la solución y el costo computacional. Si bien una población más grande (500 individuos) permite una mayor exploración del espacio de soluciones, también incrementa significativamente el tiempo de ejecución de cada generación. Se observó que el aumento del tamaño de la población a 500 individuos no producía una mejora sustancial en el costo final de la red que justificara el considerable aumento en el tiempo de cómputo. La configuración de 200 individuos demostró ser suficiente para mantener la diversidad genética y encontrar soluciones de alta calidad de manera más eficiente, representando un mejor compromiso entre la eficacia del algoritmo y los recursos computacionales disponibles.

Una vez identificados los valores de mejor desempeño para las variables tamaño de población y cantidad de generaciones, estos se fijaron y se pasó a explorar las combinaciones restantes entre p_c y p_m. Las configuraciones analizadas se presentan en la tabla 5.2. La configuración paramétrica se realizó sobre un escenario más reducido en comparación con los escenarios utilizados para la evaluación, los cuales se presentan más adelante en la sección 5.2.1. Usar un escenario de prueba más chico permitió acortar significativamente los tiempos de ejecución sin perder la validez comparativa entre configuraciones. Este procedimiento permitió identificar la configuración más adecuada para el algoritmo en función de los resultados obtenidos a partir de las pruebas realizadas.

Configuración	Tamaño de población	Cantidad de generaciones	Prob. de Cruzamiento	Prob. de Mutación
POB_100_GEN_200	100	200	0,75	0,05
POB_200_GEN_200	200	200	0,75	0,05
POB_500_GEN_200	500	200	0,75	0,05
POB_100_GEN_500	100	500	0,75	0,05
POB_200_GEN_500	200	500	0,75	0,05
POB_500_GEN_500	500	500	0,75	0,05
POB_100_GEN_1000	100	1000	0,75	0,05
POB_200_GEN_1000	200	1000	0,75	0,05
POB_500_GEN_1000	500	1000	0,75	0,05

Tabla 5.1: Configuraciones utilizadas para evaluar el tamaño de población y número de generaciones.

Configuración	Tamaño de población	Cantidad de generaciones	Prob. de Cruzamiento	Prob. de Mutación
CRUZ_6_MUT_1	200	500	0,6	0,1
CRUZ_6_MUT_01	200	500	0,6	0,01
CRUZ_6_MUT_05	200	500	0,6	0,05
CRUZ_9_MUT_1	200	500	0,9	0,1
CRUZ_9_MUT_01	200	500	0,9	0,01
CRUZ_9_MUT_05	200	500	0,9	0,05
CRUZ_75_MUT_1	200	500	0,75	0,1
CRUZ_75_MUT_01	200	500	0,75	0,01
CRUZ_75_MUT_05	200	500	0,75	0,05

Tabla 5.2: Configuraciones utilizadas para evaluar las probabilidades de cruzamiento y mutación.

5.1.2.1. Procedimientos estadísticos para la configuración paramétrica

Con el objetivo de determinar si existen diferencias significativas entre los resultados obtenidos por las distintas configuraciones del algoritmo evolutivo, se aplicaron diversos análisis estadísticos.

En primer lugar, se evaluó si los datos correspondientes al costo mínimo alcanzado por cada configuración seguían una distribución normal. Para ello, se utilizó el test de Kolmogorov-Smirnov (KS) sobre los resultados de cada conjunto de 50 ejecuciones. El test contrastó la hipótesis nula de que los datos provenían de una distribución normal, considerándose un valor-p mayor a 0,05 como indicativo de normalidad. Los valores obtenidos fueron:

- `CRUZ_6_MUT_1`: $p = 0{,}14$
- `CRUZ_6_MUT_01`: $p = 0{,}094$
- `CRUZ_6_MUT_05`: $p = 0{,}56$
- `CRUZ_9_MUT_1`: $p = 0{,}35$
- `CRUZ_9_MUT_01`: $p = 0{,}83$
- `CRUZ_9_MUT_05`: $p = 0{,}38$
- `CRUZ_75_MUT_1`: $p = 0{,}59$
- `CRUZ_75_MUT_01`: $p = 0{,}41$
- `CRUZ_75_MUT_05`: $p = 0{,}85$

En todos los casos, los valores-p fueron mayores a 0,05, lo cual indicó que no se encontró evidencia estadísticamente significativa para rechazar la hipótesis

de normalidad. Por tanto, se asume que los resultados del fitness para cada configuración provenían de una distribución normal, lo que habilitó el uso de tests paramétricos en etapas posteriores del análisis.

Luego, se aplicó la prueba de Levene para evaluar la homogeneidad de las varianzas entre configuraciones. El resultado ($p = 8{,}17 \times 10^{-12}$) indicó heterogeneidad de varianzas, lo que llevó a utilizar el test no paramétrico de Kruskal-Wallis en lugar de un ANOVA. El test de Kruskal-Wallis resultó en un valor $p < 0,001$, indicando diferencias estadísticamente significativas entre al menos dos configuraciones.

Para identificar qué configuraciones diferían entre sí, se aplicó un test post-hoc de Dunn con corrección de Holm para controlar el error tipo I. Los resultados se presentan en la tabla 5.3.

La tabla 5.3 muestra que la configuración `CRUZ_75_MUT_05` presentó diferencias estadísticamente significativas con la mayoría de las demás configuraciones, lo que reforzó su buen desempeño.

	CRUZ_6_MUT_01	CRUZ_6_MUT_05	CRUZ_6_MUT_1	CRUZ_75_MUT_01	CRUZ_75_MUT_05
CRUZ_6_MUT_01	1,000	0,344	$2{,}96 \times 10^{-17}$	$2{,}71 \times 10^{-4}$	$1{,}92 \times 10^{-27}$
CRUZ_6_MUT_05		1,000	$1{,}08 \times 10^{-10}$	0,173	$5{,}99 \times 10^{-19}$
CRUZ_6_MUT_1			1,000	$1{,}07 \times 10^{-4}$	0,173
CRUZ_75_MUT_01				1,000	$1{,}81 \times 10^{-10}$
CRUZ_75_MUT_05					1,000

(a) Comparaciones entre las primeras cinco configuraciones

	CRUZ_75_MUT_1	CRUZ_9_MUT_01	CRUZ_9_MUT_05	CRUZ_9_MUT_1
CRUZ_6_MUT_01	0,080	0,956	$2{,}52 \times 10^{-12}$	$2{,}58 \times 10^{-9}$
CRUZ_6_MUT_05	0,956	0,173	$6{,}25 \times 10^{-7}$	$9{,}66 \times 10^{-5}$
CRUZ_6_MUT_1	$2{,}10 \times 10^{-8}$	$4{,}97 \times 10^{-19}$	0,665	0,173
CRUZ_75_MUT_01	0,545	$4{,}10 \times 10^{-5}$	$2{,}17 \times 10^{-2}$	0,223
CRUZ_75_MUT_05	$8{,}24 \times 10^{-16}$	$1{,}10 \times 10^{-29}$	$3{,}06 \times 10^{-3}$	$4{,}92 \times 10^{-5}$
CRUZ_75_MUT_1	1,000	$2{,}17 \times 10^{-2}$	$4{,}34 \times 10^{-5}$	$2{,}84 \times 10^{-3}$
CRUZ_9_MUT_01		1,000	$7{,}62 \times 10^{-14}$	$1{,}27 \times 10^{-10}$
CRUZ_9_MUT_05			1,000	0,956
CRUZ_9_MUT_1				1,000

(b) Comparaciones con configuraciones restantes

Tabla 5.3: Resultados del test de Dunn con corrección de Holm (valores-p).

5.1.3. Resultados de la configuración paramétrica

La tabla 5.4 presenta un resumen comparativo del desempeño de cada configuración evaluada. La tabla incluye los valores mínimo, promedio y máximo del costo de la red obtenido, así como la desviación estándar, el tiempo promedio de ejecución y la cantidad promedio de generaciones para alcanzar el costo mínimo. Estos indicadores permitieron visualizar rápidamente el comportamiento general de cada configuración, destacándose `CRUZ_75_MUT_05` por haber logrado el menor valor promedio del costo total de la red con una de las desviaciones más bajas, lo cual sugirió una buena combinación de eficiencia y estabilidad. También se registraron diferencias importantes en el tiempo de ejecución y número de generaciones necesarias para alcanzar el costo mínimo.

Configuración	Costo mínimo	Costo promedio	Costo máximo	Desviación estándar	Duración (segundos)	Generaciones min costo
CRUZ_75_MUT_05	2.512.211	2.602.789	2.686.958	38.057	1.276	423,50
CRUZ_75_MUT_1	2.622.211	2.753.151	2.834.055	45.357	2.133	424,56
CRUZ_75_MUT_01	2.636.519	2.727.995	2.821.317	44.624	1.281	388,50
CRUZ_6_MUT_05	2.635.389	2.765.363	2.849.845	57.817	1.032	380,84
CRUZ_6_MUT_1	2.537.057	2.647.958	2.776.283	59.277	877	412,96
CRUZ_6_MUT_01	2.591.692	2.807.754	2.948.758	89.522	964	374,96
CRUZ_9_MUT_05	2.595.310	2.676.404	2.752.992	43.979	2.202	431,36
CRUZ_9_MUT_1	2.559.845	2.689.232	2.804.228	77.175	2.362	443,28
CRUZ_9_MUT_01	2.726.410	2.803.408	2.877.739	36.311	2.086	429,92

Tabla 5.4: Resumen estadístico de desempeño por configuración.

Para complementar el análisis estadístico, se construyó un gráfico de barras que muestra el costo promedio de cada configuración junto con su desviación estándar. Esta visualización ilustra de forma rápida cuáles configuraciones obtuvieron mejores resultados y con qué nivel de consistencia. La figura 5.1 presenta los valores promedio de costo obtenidos por cada configuración del algoritmo, junto con sus respectivas desviaciones estándar. La configuración `CRUZ_75_MUT_05` no solo alcanzó el menor costo promedio, sino que también presentó una baja variabilidad, resultando en una buena estabilidad del algoritmo bajo dicha parametrización. En contraposición, otras configuraciones como `CRUZ_6_MUT_01` y `CRUZ_9_MUT_1` presentaron mayor dispersión en los resultados, lo que indica un comportamiento menos consistente. Esta visualización permitió reforzar la conclusión de que la configuración seleccionada no solo es eficiente, sino también robusta.

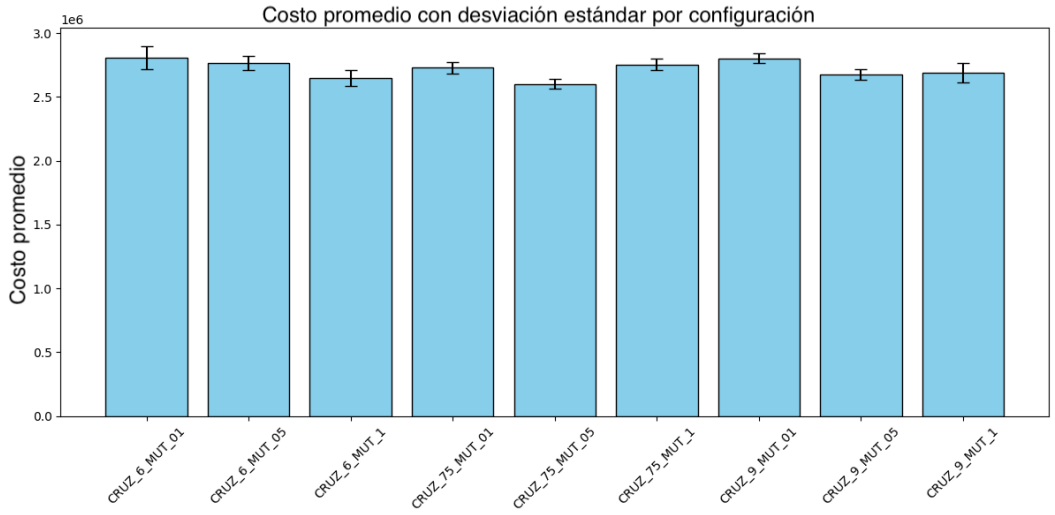


Figura 5.1: Promedios y desviaciones estándar del costo por configuración.

En la figura 5.2 se presenta la distribución de los valores del costo total mínimo para cada configuración mediante diagramas de caja. Esta visualización muestra la tendencia central y la dispersión de los resultados. Se destaca que la configuración `CRUZ_75_MUT_05` obtuvo, en promedio, los mejores valores de costo y presentó una variabilidad relativamente acotada.

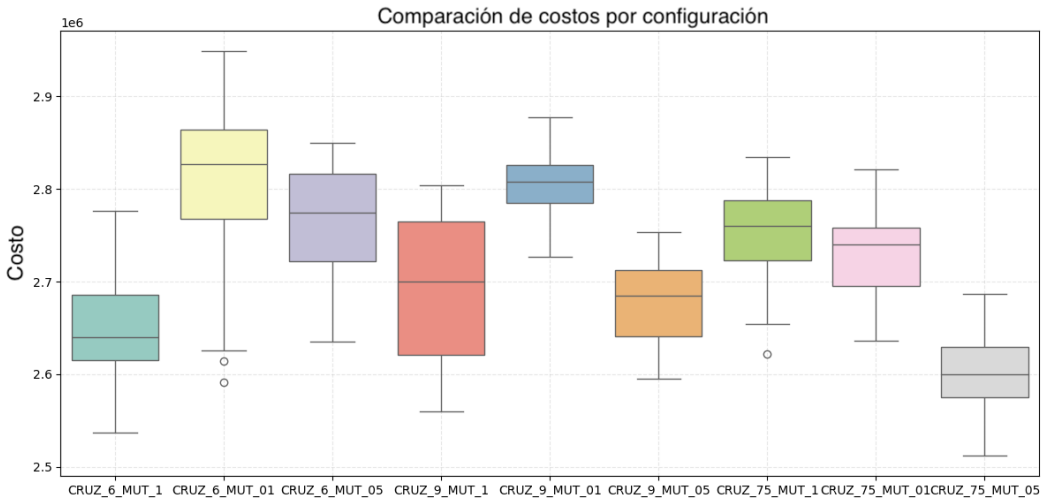


Figura 5.2: Distribución de valores de los costos mínimos obtenidos para cada configuración del algoritmo.

La configuración `CRUZ_75_MUT_05` se consolidó como la mejor configuración evaluada, tanto por su posición destacada en el ranking combinado como por las diferencias estadísticamente significativas con la mayoría de sus pares, lo

que brindó evidencia robusta para seleccionarla como la configuración que mejor se adaptó en este contexto.

5.1.4. Selección de la configuración paramétrica

A partir de los análisis realizados, se concluyó que la configuración `CRUZ_75_MUT_05` presentó el mejor desempeño general. Esta afirmación se basó en tres criterios complementarios: obtuvo el menor costo promedio (figura 5.2), registró una de las desviaciones estándar más bajas (figura 5.1), y mostró diferencias estadísticamente significativas respecto a la mayoría de las otras configuraciones (tabla 5.3).

La coherencia entre estos criterios reforzó la confiabilidad de la elección. En particular, la baja variabilidad asociada a esta configuración sugirió un comportamiento robusto, y los resultados del test de Dunn permitieron descartar que su mejor rendimiento fuera producto del azar.

Por lo tanto, se seleccionó `CRUZ_75_MUT_05` como la configuración de mejor desempeño para utilizar en el análisis experimental completo del algoritmo. Esta configuración correspondió a una probabilidad de cruzamiento de 0,75, una probabilidad de mutación de 0,05, tamaño de población de 200 y número de generaciones igual a 500.

5.2. Evaluación del algoritmo evolutivo

Esta sección evalúa el desempeño del algoritmo sobre un conjunto representativo de instancias del problema de diseño de redes de saneamiento. El objetivo principal de esta etapa es analizar la calidad de las soluciones generadas, su consistencia entre ejecuciones, y la escalabilidad del enfoque propuesto frente a instancias de diferente tamaño y complejidad.

Se definió un conjunto de instancias de prueba que abarcaron diversas características topológicas y demandas de caudal. Cada instancia fue resuelta mediante 50 ejecuciones independientes del algoritmo con la configuración seleccionada en la configuración paramétrica, lo que permitió obtener estadísticas descriptivas como el valor del costo promedio, la desviación estándar, el mejor y peor valor de costo alcanzado, y otras métricas relevantes para el análisis.

En esta sección se presentan y discuten los resultados obtenidos, incluyendo tablas de rendimiento, gráficos de dispersión y boxplots que permiten visua-

lizar el comportamiento global del algoritmo. Asimismo, se exploran aspectos específicos como la convergencia del proceso evolutivo, la variabilidad entre ejecuciones y la eficiencia computacional.

5.2.1. Instancias utilizadas en la evaluación

Para evaluar el desempeño del algoritmo evolutivo, se diseñaron tres escenarios de prueba con características distintas, basados en datos reales de la ciudad de Montevideo. Las propiedades de cada una de estas instancias se resumieron en la tabla 5.5. A continuación, se describe cada escenario en detalle.

Característica	Escenario 1	Escenario 2	Escenario 3
Nº de vértices (Esquinas)	182	122	197
Nº de aristas (Calles)	297	187	292
Área (km^2)	1,6	1,07	1,8
Población estimada	16407	23013	19385
Rango de alturas (msnm)	32 - 70	14 - 44	32 - 58
Longitud promedio calle (m)	93	99	98

Tabla 5.5: Características de los escenarios utilizados.

5.2.1.1. Escenario 1

El primer escenario utilizado para la evaluación experimental correspondió a la red vial del barrio Cerrito de la Victoria en Montevideo, un barrio residencial conocido por su distintiva topografía. La elección de este escenario, que cubre un área de 1,6 km^2, se debió a la interesante combinación de un trazado vial que combina sectores en cuadrícula, una densidad poblacional media para la ciudad (con una población estimada de 16.407 habitantes), y fundamentalmente, variaciones de altura significativas. El grafo resultante consistió en 182 vértices y 297 aristas, con longitudes de tramo que oscilaron entre los 20 y los 211 metros.

La característica topográfica principal fue el cerrito que da nombre al barrio, el cual generó pendientes pronunciadas en sus laderas y zonas más planas en su base y cima. Las alturas de las esquinas variaron entre 32 msnm y 70 msnm, como se visualiza en el mapa topográfico de la figura 5.3.

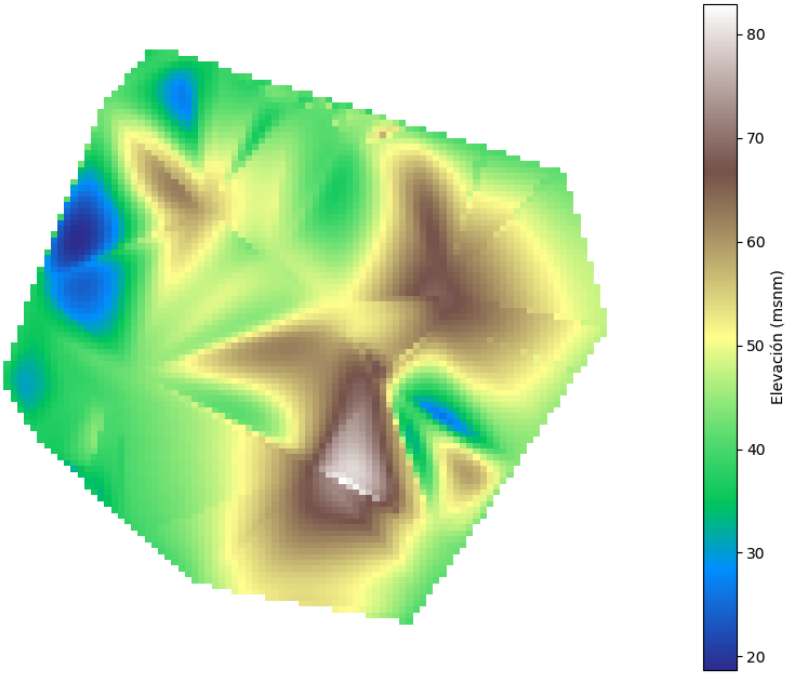


Figura 5.3: Mapa topográfico del escenario 1

La trama de calles del escenario se presenta en la figura 5.4, donde el nodo sumidero global se marca de color rojo. El nodo sumidero se ubicó en el punto de menor altura del escenario, en las coordenadas (-34.85°,-56.17°). Este conjunto de características representó un desafío relevante para el diseño de una red gravitacional eficiente.

5.2.1.2. Escenario 2

El segundo escenario abarcó una zona híbrida que incluye la mitad sur del barrio Parque Batlle y la mitad norte del barrio Pocitos. Se trató de un escenario urbano que combina la alta densidad de un barrio como Pocitos, con la estructura abierta e irregular de un parque urbano como Parque Batlle. Se diseñó este escenario para evaluar la capacidad de adaptación del algoritmo a características urbanas distintas dentro de una misma red. El objetivo fue analizar cómo el algoritmo gestiona la transición entre una zona de alta densidad y demanda (Pocitos) y una zona de menor densidad (Parque Batlle).

El grafo que representó al escenario 2, cubrió un área total de 1,07 km^2 y constó de 122 vértices y 187 aristas. La población total estimada a atender

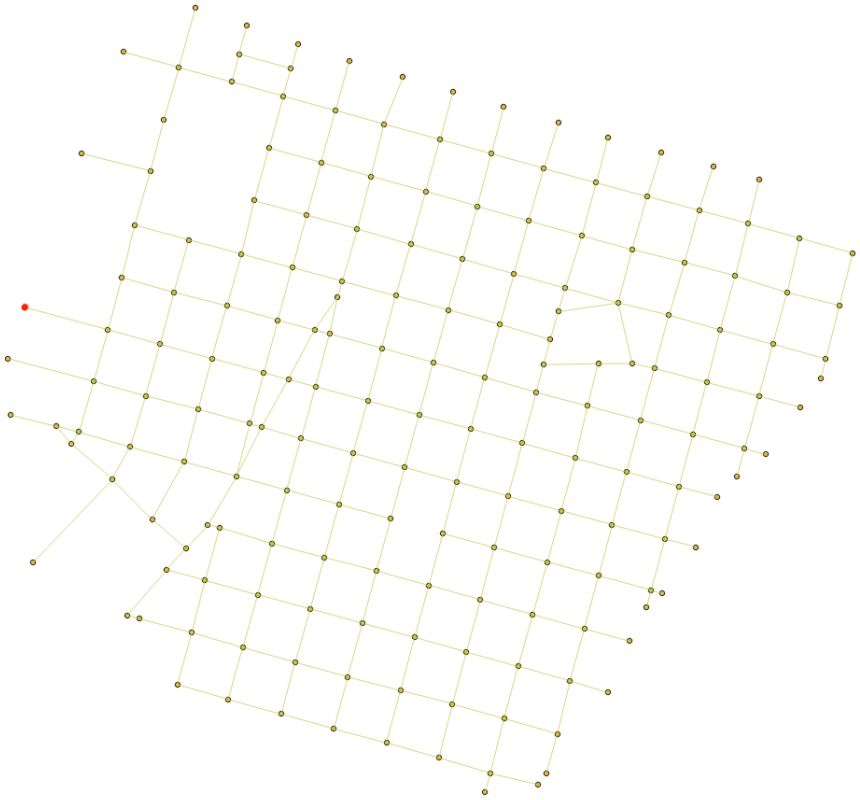


Figura 5.4: Trama de calles del escenario 1

fue de 23.013 habitantes. La topografía del escenario fue relativamente suave, con una pendiente general constante desde la zona de Parque Batlle hacia la costa en Pocitos. Las alturas variaron entre 44 msnm en el extremo norte del escenario y 14 msnm en el extremo sur. La longitud de las aristas varió entre un mínimo de 20 metros y un máximo de 209 metros, siendo 99 metros la longitud promedio.

La sección norte, correspondiente a Parque Batlle, se caracterizó por su estructura abierta e irregular que rodea el parque. Por otro lado, la sección sur (Pocitos), si bien fue densa, no siguió una trama en cuadrícula. En su lugar, presentó una red vial compleja con calles diagonales y bloques de formas irregulares, como se muestra en la figura 5.5.

En la figura 5.6 se presenta el mapa topográfico del escenario, el cual presentó una pendiente suave pero constante en dirección norte-sur. Esta característica resultó favorable en el diseño de la red gravitacional. La demanda fue muy alta en la mitad correspondiente a Pocitos, y más baja y dispersa en la

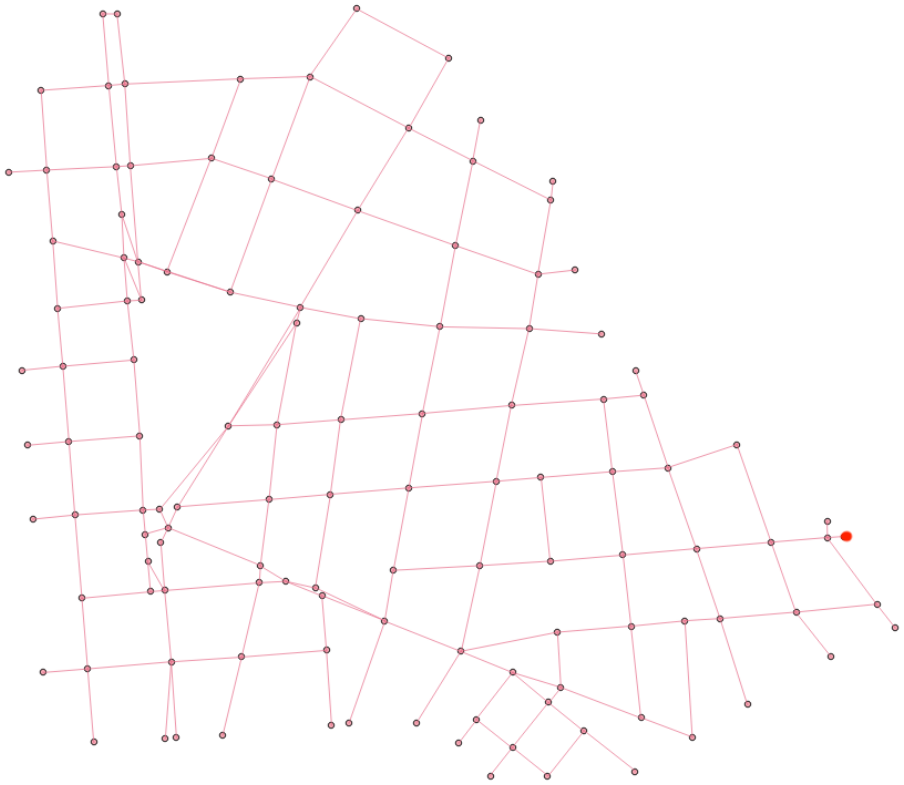


Figura 5.5: Trama de calles del escenario 2

mitad correspondiente a Parque Batlle. En este escenario, el nodo sumidero global se ubicó en las coordenadas (-34,90°, -56,15°).

5.2.1.3. Escenario 3

El tercer escenario corresponde al barrio La Unión. La Unión es conocida por ser un importante centro comercial principalmente a lo largo de la Avenida 8 de Octubre y por una sección residencial en sus alrededores.

El escenario 3 fue escogido para evaluar el desempeño del algoritmo en un contexto de alta demanda y, fundamentalmente, en una topografía predominantemente llana. Este último factor introdujo el desafío de diseñar una red gravitacional eficiente con pendientes muy suaves, lo que puso a prueba la capacidad del algoritmo para minimizar los costos de excavación sin comprometer la factibilidad hidráulica.

El grafo representativo del escenario 3 contó con 197 vértices y 292 aristas y cubrió un área aproximada de 1,8 km^2. La población estimada dentro del

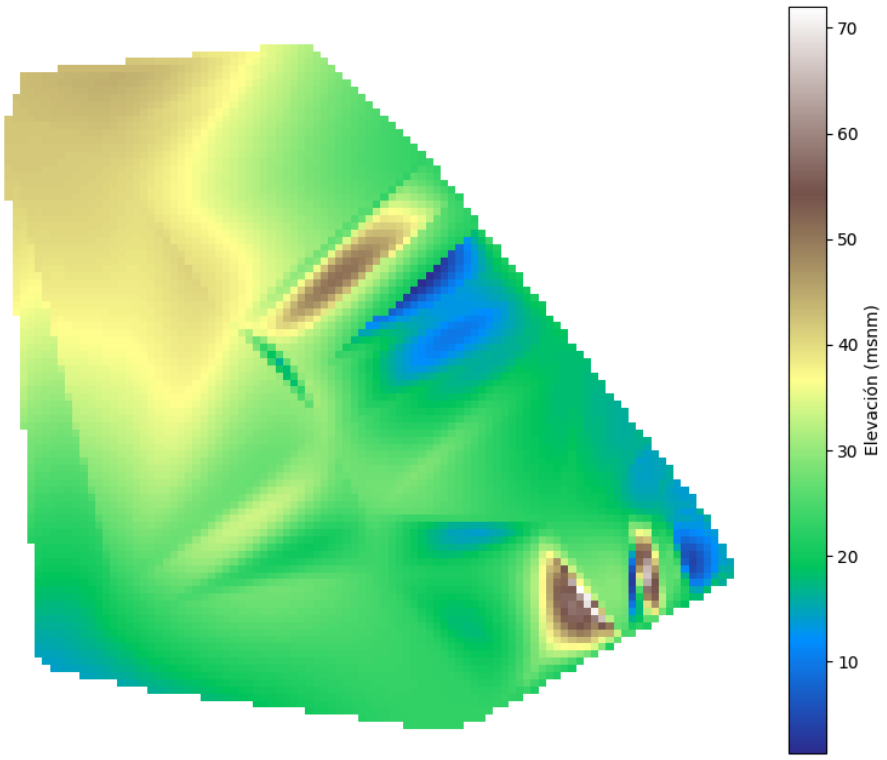


Figura 5.6: Mapa topográfico del escenario 2

área modelada fue de 19.385 habitantes. La topografía fue mayormente plana, con variaciones de altura mínimas. Las cotas se encontraron en un rango de 32msnm y 58msnm, presentando un desafío para el diseño por gravedad.

El trazado vial corresponde a una cuadrícula bastante regular. Presenta una alta densidad y a diferencia de los otros escenarios evaluados, la Unión se caracteriza por un terreno predominante llano, con pendientes naturales muy leves. Esta es su principal característica distintiva desde el punto de vista del diseño de redes de saneamiento. El nodo sumidero para este escenario se ubicó en las coordenadas (-34,88°, -56,12°). La figura 5.7 muestra el trazado de calles del escenario y la figura 5.8 muestra el mapa topográfico del escenario.

5.2.2. Evolución del fitness

En esta sección se analiza el comportamiento dinámico del algoritmo a lo largo de las generaciones en los distintos escenarios de prueba.

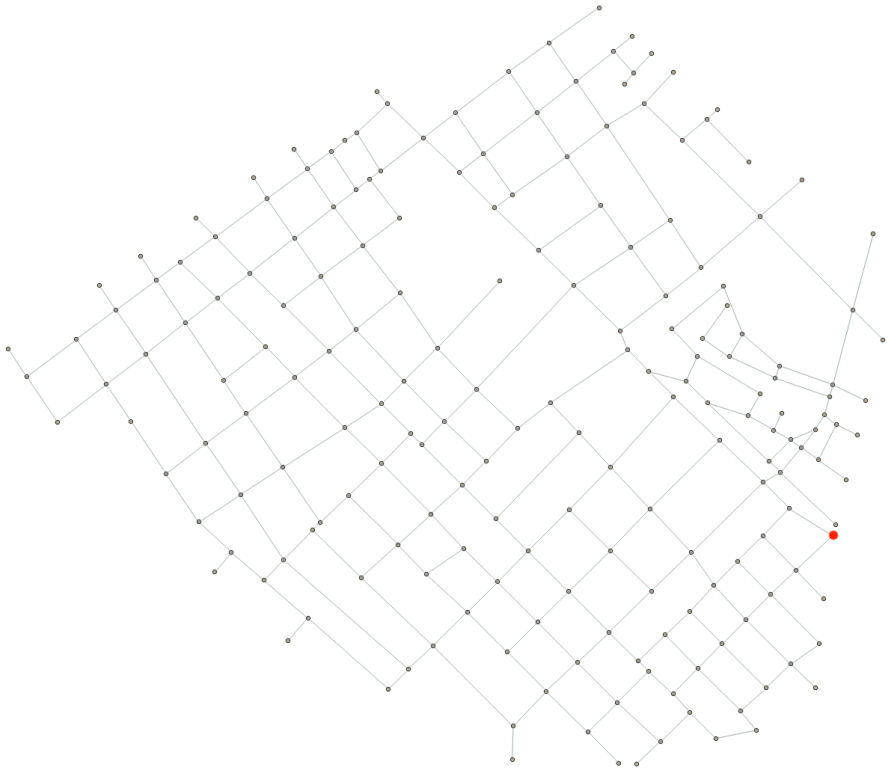


Figura 5.7: Trama de calles del escenario 3

Las figuras 5.9, 5.10 y 5.11 muestran la evolución del costo mínimo de la red en múltiples ejecuciones. Si bien se realizaron 50 ejecuciones para cada escenario, con el objetivo de mantener la figuras claras y legibles, se decidió crearlas con 10 generaciones representativas.

Para los tres escenarios, las figuras 5.9, 5.10 y 5.11 muestran un patrón consistente: una disminución pronunciada del costo de la red en las primeras generaciones, seguida por una etapa de estabilización progresiva, lo que refleja la capacidad del algoritmo para mejorar progresivamente las soluciones generadas. La reducción más significativa del costo ocurrió durante las primeras 100 generaciones, lo cual es esperable en este tipo de métodos, dado que las mayores mejoras suelen lograrse en las etapas iniciales de la evolución. A medida que avanzó el proceso, las curvas tendieron a estabilizarse, indicando una fase de convergencia.

La visualización del costo mínimo evidencia la variabilidad entre ejecuciones, mientras que la evolución del promedio junto con la banda de desviación estándar permite evaluar la estabilidad y consistencia del desempeño. La figura 5.12 resume el comportamiento típico del algoritmo en conjunto, destacando

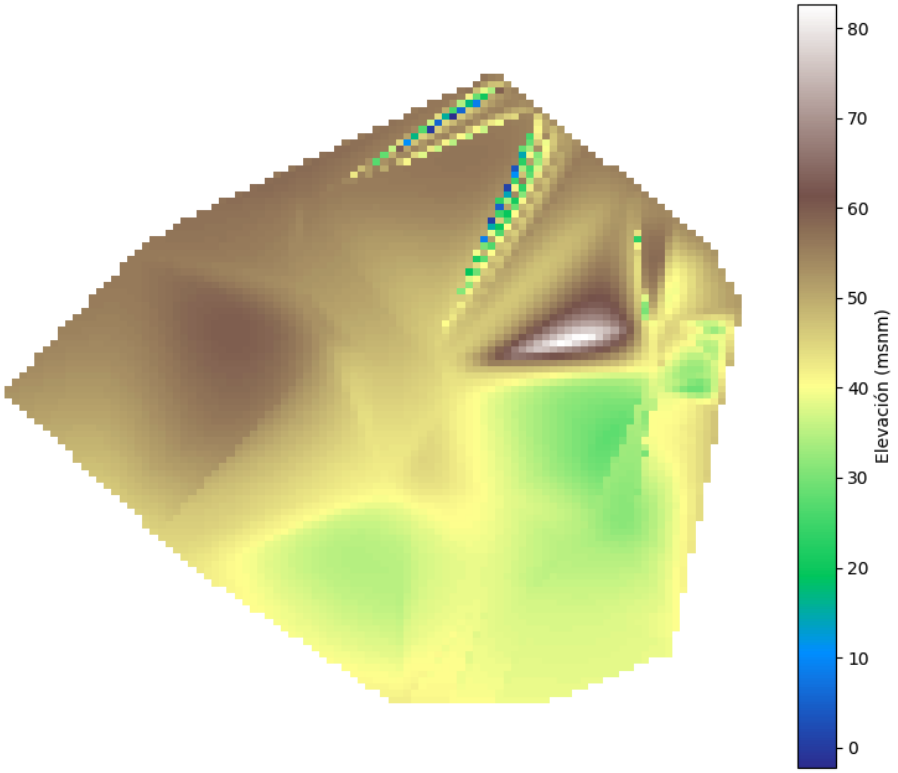


Figura 5.8: Mapa topográfico del escenario 3

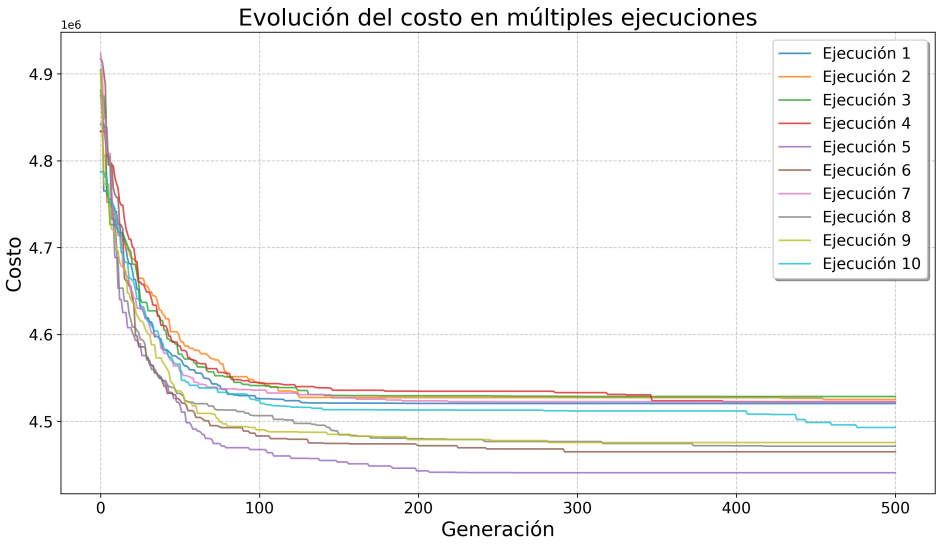


Figura 5.9: Evolución del costo para 10 ejecuciones del escenario 1

tanto su desempeño medio como su estabilidad entre ejecuciones. Para los tres casos, existió una mejora sostenida del fitness hasta cerca de la generación 200,

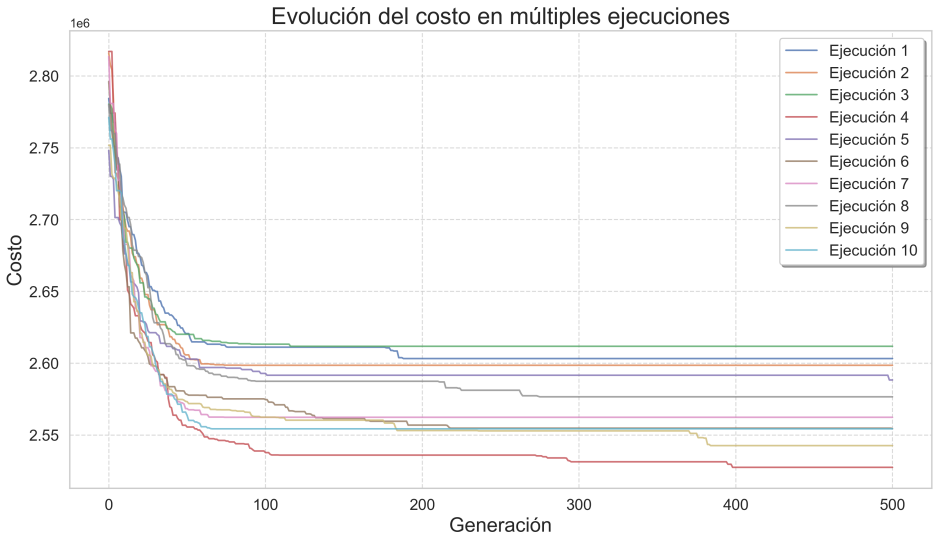


Figura 5.10: Evolución del costo para 10 ejecuciones del escenario 2

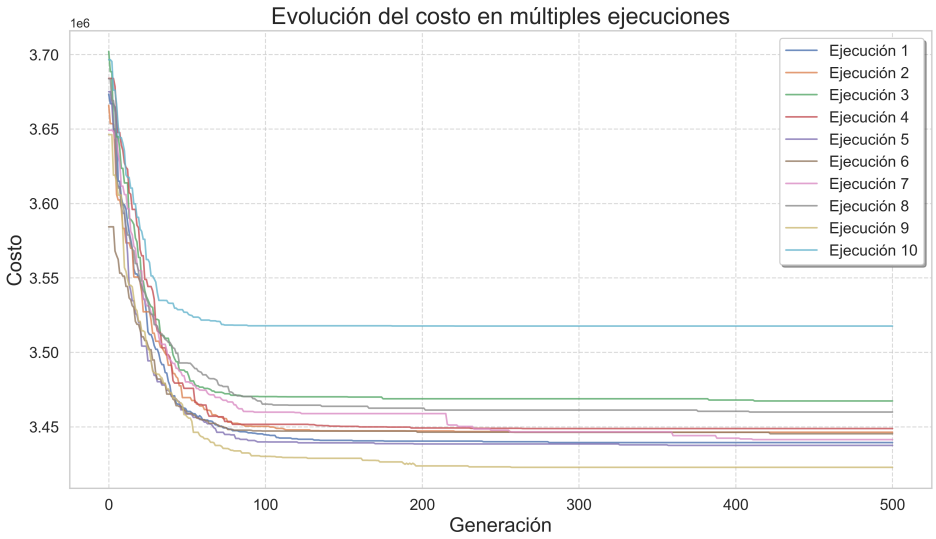


Figura 5.11: Evolución del costo para 10 ejecuciones del escenario 3

a partir de la cual la evolución se ralentizó. La amplitud de la banda de desviación estándar muestra que, si bien existió variabilidad entre ejecuciones, esta se mantuvo dentro de márgenes razonables, lo que sugirió un comportamiento robusto del algoritmo bajo la configuración seleccionada. Aunque existieron diferencias en la magnitud de los valores debido a las particularidades de cada escenario, el comportamiento general del algoritmo fue estable y predecible en todos los casos. Estos resultados reforzaron la robustez de la configuración

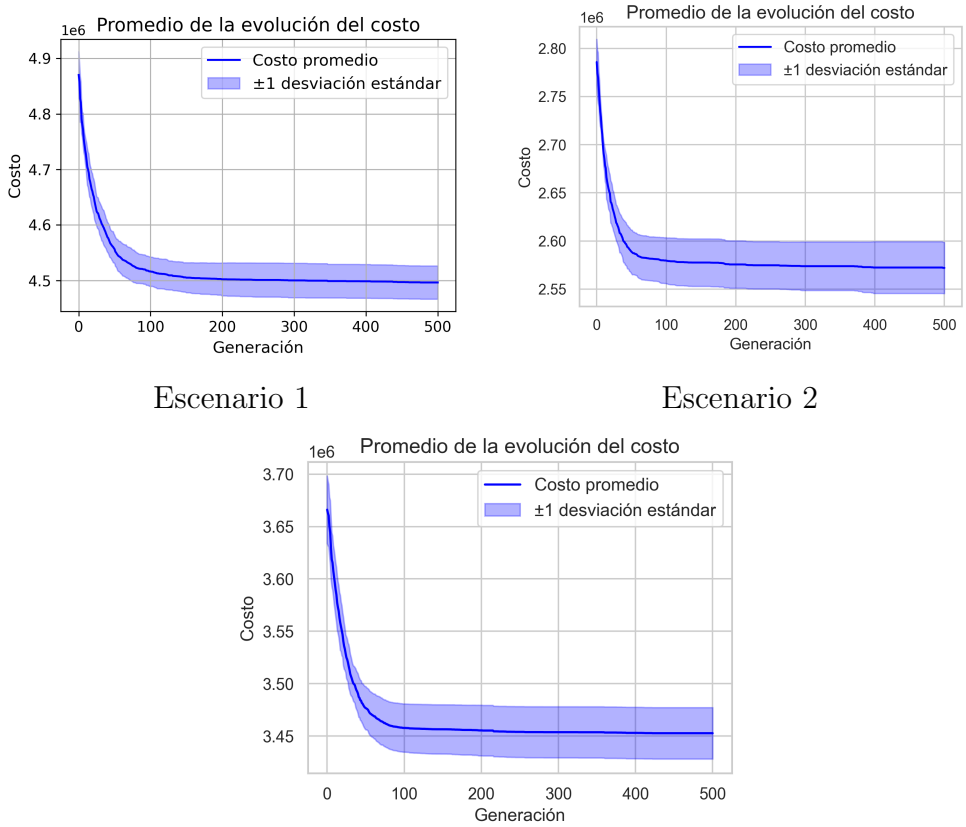


Figura 5.12: Evolución del costo de las 50 generaciones en promedio con la desviación estándar en los distintos escenarios de prueba.

paramétrica seleccionada, mostrando que el algoritmo logró mantener un buen desempeño aun en escenarios de prueba con características distintas.

5.2.3. Normalidad del fitness

Con el objetivo de analizar la distribución estadística de los resultados obtenidos por el algoritmo evolutivo, se aplicó el test de Kolmogorov–Smirnov a los valores de fitness final alcanzados en las 50 ejecuciones correspondientes a cada instancia de prueba. Este test permite contrastar la hipótesis nula de que los datos provienen de una distribución normal.

Los resultados obtenidos se presentan a continuación:

- ESCENARIO_1: estadístico KS = 0,076, valor-p = 0,916
- ESCENARIO_2: estadístico KS = 0,095, valor-p = 0,720
- ESCENARIO_3: estadístico KS = 0,156, valor-p = 0,157

En los tres escenarios, los valores-p fueron considerablemente mayores que el umbral habitual de $\alpha = 0{,}05$, por lo que no se rechazó la hipótesis nula de normalidad. Esto sugirió que los resultados obtenidos por el algoritmo en cada escenario pudieron considerarse provenientes de una distribución normal, lo cual fue tenido en cuenta en los análisis comparativos posteriores contra algoritmos de referencia.

5.2.4. Desempeño por escenario

En la tabla 5.6 se presentan los principales indicadores de desempeño obtenidos a partir de las 50 ejecuciones del algoritmo evolutivo sobre cada instancia de prueba. Se incluye el valor promedio de mejor costo alcanzado, así como la desviación estándar, el tiempo promedio de ejecución y la cantidad promedio de iteraciones que le tomó al algoritmo en encontrar la solución de menor costo.

Instancia	Fitness	Duración	Iteraciones	Desviación estándar
ESCENARIO_1	4.488.080,59	20.234 s	310,65	36.848,36
ESCENARIO_2	2.565.077,44	11.634 s	224,15	27.421,70
ESCENARIO_3	3.450.248,11	19.445 s	285,8	21.814,76

Tabla 5.6: Resultados de desempeño del algoritmo evolutivo en las tres instancias de prueba.

En términos generales, el algoritmo mostró un comportamiento robusto en los tres escenarios, con valores de fitness promedio estables y una variabilidad controlada, tal como lo indicaron las desviaciones estándar moderadas.

Para el ESCENARIO_1, se alcanzó un costo mínimo de 4.420.724,84, siendo el promedio de 4.488.080,59 y una desviación estándar de 36.848,36. Esto sugirió una ligera mayor dispersión en los resultados, aunque sin comprometer la estabilidad del algoritmo. Las ejecuciones demoraron en promedio 20.234 segundos, completando cerca de 310 generaciones en promedio para encontrar la solución de costo mínimo.

En ESCENARIO_2, el algoritmo mostró resultados más concentrados: la mejor solución tuvo un costo de 2.520.850,2 y el promedio fue de 2.565.077,44, con una desviación estándar menor (27.421,70) y un tiempo promedio de ejecución de 11.634 segundos. En este caso, el número promedio de generaciones

hasta llegar al costo mínimo fue inferior, alcanzando el mínimo en promedio en 224 iteraciones.

Por último, en ESCENARIO_3, el valor mínimo registrado fue de 3.409.582,95, con un promedio de 3.450.248,11 y una desviación estándar de 21.814,76. Las ejecuciones fueron más prolongadas en promedio, con una duración de 19.445 segundos y cerca de 285 iteraciones por corrida para llegar a la mejor solución.

Estos resultados evidenciaron que el algoritmo mantuvo un rendimiento consistente en distintas condiciones de prueba, tanto en términos de calidad de soluciones como de estabilidad entre ejecuciones.

5.3. Comparación con algoritmo de referencia

Como parte de la evaluación del desempeño del algoritmo evolutivo, se lo comparó con un algoritmo greedy determinístico sobre las tres instancias de prueba. El algoritmo greedy genera siempre la misma solución, por lo que se obtuvo un único valor del costo por escenario, mientras que el AE fue ejecutado 50 veces con la configuración previamente definida en la sección 5.1.4.

En la tabla 5.7 se reportan para cada escenario las siguientes métricas: el costo de la solución del algoritmo greedy, el valor promedio de fitness generado por el AE, su desviación estándar, y el porcentaje de mejora con respecto al algoritmo greedy. Además, se incluye el valor-p del test de Mann–Whitney U aplicado para validar la diferencia entre ambos enfoques.

Instancia	Greedy	Prom. AE	% Mejora	p Mann–Whitney
ESCENARIO_1	5.126.891	4.488.080	12,46 %	$< 10^{-9}$
ESCENARIO_2	2.960.490	2.565.077	13,36 %	$< 10^{-9}$
ESCENARIO_3	3.862.356	3.450.248	10,67 %	$< 10^{-9}$

Tabla 5.7: Comparación entre el algoritmo evolutivo y el algoritmo greedy.

El porcentaje de mejora se calculó según la siguiente expresión:

$$\%\text{mejora} = \frac{f(\text{Greedy}) - f(\text{AE})}{f(\text{Greedy})} \times 100$$

Para validar estadísticamente esta diferencia, se aplicó el test no paramétrico de Mann–Whitney U propuesto por Mann y Whitney (1947) entre las eje-

cuciones del AE y el valor constante del algoritmo greedy (replicado 50 veces). En las tres instancias, el valor p fue inferior a 10^{-9}, lo que proporcionó evidencia sólida de que las soluciones obtenidas por el AE fueron significativamente mejores que las del algoritmo greedy.

El test de Mann–Whitney U se selecciona por ser particularmente adecuado para la comparación entre el algoritmo evolutivo y el algoritmo greedy, ya que el algoritmo evolutivo es de naturaleza estocástica y produce una distribución de valores de fitness, mientras que el algoritmo greedy es determinístico y siempre genera una única solución. En este contexto, no fue apropiado utilizar tests como Student t-test o ANOVA, que requirieron que ambas muestras tuvieran varianza y siguieran distribuciones normales. El test de Mann–Whitney U, al no requerir supuestos sobre la distribución ni sobre las varianzas, resulta robusto y confiable para comparar una muestra aleatoria con un valor fijo replicado.

De forma complementaria, se aplicó una regla heurística que sugiere que un algoritmo A puede considerarse significativamente mejor que B si la diferencia de sus valores promedio supera la mayor desviación estándar.

$$|\bar{f}_{AE} - f_{Greedy}| > \text{máx}(\sigma_{AE}, 0),$$

En este caso, como el algoritmo greedy no tuvo varianza, se verificó si la diferencia entre la media del AE y el valor del algoritmo greedy fue mayor que la desviación estándar del AE. Este criterio se cumplió ampliamente en los tres escenarios, reforzando la conclusión de que el algoritmo evolutivo presentó un desempeño claramente superior al algoritmo greedy.

La figura 5.13 muestra la distribución de los valores de costo obtenidos por el AE para cada instancia, representada mediante un diagrama de caja. En cada gráfico, se incluye una línea punteada roja indicando el valor de costo alcanzado por el algoritmo greedy.

Los resultados muestran que el AE superó ampliamente al algoritmo greedy en todos los casos: incluso las peores ejecuciones del AE resultaron mejores que la solución encontrada por el algoritmo greedy. Además, mostró una consistencia destacable en los resultados del AE, con desviaciones estándar moderadas y valores promedio significativamente más bajos.

El análisis estadístico mediante el test de Mann–Whitney U confirmó que las diferencias no fueron producto del azar, sino que fueron estadísticamente

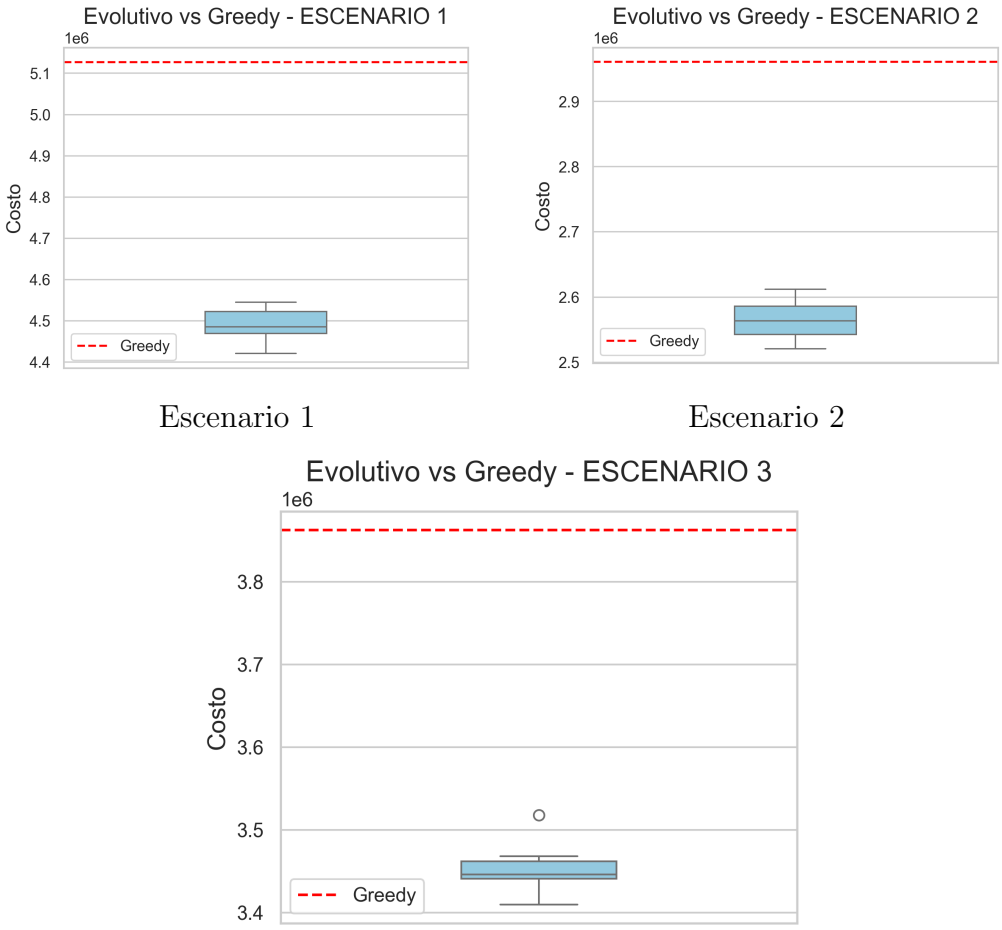


Figura 5.13: Distribución del costo final del AE comparado con el valor obtenido por el algoritmo greedy.

significativas en todos los casos. Además, las representaciones gráficas reforzaron visualmente la ventaja del AE, que incluso en sus peores ejecuciones superó a la única solución generada por el algoritmo greedy.

5.3.1. Análisis cualitativo y estructural de las soluciones

Con el objetivo de comprender en mayor profundidad las diferencias entre las soluciones generadas por el algoritmo greedy y el algoritmo evolutivo, se realizó un análisis cualitativo centrado en tres aspectos clave: el volumen total de excavación, la distribución de los diámetros utilizados y los trazados de red generados. Para este estudio, se tomó una única ejecución del algoritmo

evolutivo por escenario, obteniendo una solución representativa que se comparó con la correspondiente solución del enfoque greedy.

En el caso del escenario 1, el algoritmo evolutivo generó una red con un costo total de 4.553.231,92, lo que representa una mejora del 10,49 % respecto a la solución obtenida por el algoritmo greedy, cuyo costo ascendió a 5.087.046,21. Este ahorro económico se relaciona con varios factores:

- En primer lugar, la solución evolutiva presentó una distribución más eficiente de los diámetros de tubería. En particular, se redujo considerablemente la utilización de tramos de 400 mm, que poseen un costo unitario elevado, incrementando en su lugar el uso de tramos de 150 mm y 200 mm, más acordes al caudal a transportar.
- En segundo lugar, el trazado espacial de la red resultó más eficiente, con una reducción de 413 metros en la longitud total de tuberías y una disminución de más de 13.000 m^3 en el volumen de excavación requerido, lo que implica un ahorro directo en obra civil.

Estos resultados reflejan la capacidad del algoritmo evolutivo para optimizar simultáneamente múltiples dimensiones del diseño. No solo genera soluciones factibles desde el punto de vista hidráulico y topológico, sino que también evita sobredimensionamientos innecesarios y selecciona rutas más económicas y técnicamente convenientes.

La tabla 5.8 sintetiza las métricas obtenidas para cada solución en el escenario 1. Si bien la profundidad máxima alcanzada en la solución evolutiva fue ligeramente superior, la profundidad promedio ponderada resultó menor, lo que indica un diseño general más superficial, y por tanto, más económico y sencillo de ejecutar.

Métrica	**Greedy**	**AE**	**% Mejora**
Volumen excavación (m^3)	119.875,61	106.248,06	11,37 %
Longitud total tuberías (m)	16.840,00	16.427,00	2,45 %
Profundidad máxima (m)	28,67	30,63	-6,82 %
Profundidad promedio (m)	7,12	6,47	9,14 %

Tabla 5.8: Comparativa de las soluciones del algoritmo greedy y el evolutivo para el escenario 1.

Por su parte, la tabla 5.9 detalla la distribución de los diámetros de tubería utilizados por cada solución. Se destaca que el enfoque evolutivo empleó más

longitud de tubería de 150mm y menos de 400mm, lo que evidencia un mejor dimensionamiento en función del caudal. Esta redistribución también favoreció la reducción de costos sin comprometer la capacidad hidráulica del sistema.

Distribución de diámetros	Greedy (m)	Evolutivo (m)
Longitud tubería 100 mm	8.968	8.944
Longitud tubería 150 mm	1.191	2.321
Longitud tubería 200 mm	1.120	1.498
Longitud tubería 250 mm	1.494	642
Longitud tubería 300 mm	910	1.377
Longitud tubería 400 mm	3.157	1.645

Tabla 5.9: Comparativa de la distribución de diámetros utilizados en ambas soluciones en el escenario 1.

Para tener una visualización gráfica de las diferencias entre el AE y el greedy se generó la figura 5.14, donde se presenta el perfil longitudinal de uno de los tramos en común para el escenario 1, comparando una solución obtenida mediante el AE y la del algoritmo greedy. Ambos diseños respetaron las pendientes mínimas requeridas para garantizar el flujo por gravedad, pero existieron diferencias en la profundidad de instalación de las tuberías a lo largo del tramo. Si bien las soluciones fueron similares, la solución del AE presentó una mayor regularidad en las pendientes y una menor variación en la profundidad, lo cual contribuye a una reducción de costos.

Otra forma de representar las diferencias entre una solución dada por el AE y la que generó el greedy es la figura 5.15. Las aristas en rojo corresponden a tramos comunes presentes en ambas soluciones, mientras que las verdes representan conexiones exclusivas del algoritmo greedy y las azules aquellas utilizadas únicamente por el algoritmo evolutivo. Ambos enfoques coincidieron en una proporción significativa del trazado, especialmente en los sectores periféricos de la red. No obstante, existieron algunas diferencias notorias, sobre todo en el centro y parte baja de la figura, donde cada algoritmo optó por rutas alternativas para conectar ciertos nodos. Estas discrepancias sugieren que el algoritmo evolutivo, al explorar múltiples soluciones durante su proceso de búsqueda, logró identificar trayectorias que permitieron reducir el costo total del sistema, posiblemente aprovechando mejor las pendientes naturales del terreno o el agrupamiento eficiente de caudales.

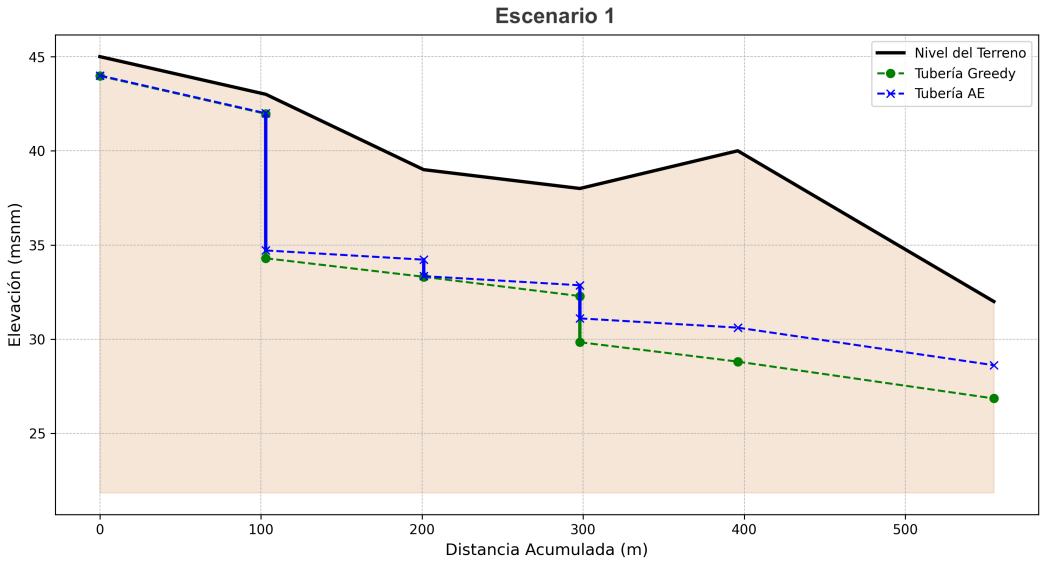


Figura 5.14: Comparación del perfil longitudinal para un camino común en el escenario 1. Se muestra el nivel del terreno (linea negra) y las cotas de las tuberías para la solución greedy (verde) y la del AE (azul)

El escenario 2 se diseñó para probar la adaptabilidad del AE a un entorno urbano híbrido, combinando la alta densidad de Pocitos con la estructura abierta de Parque Battle. Este escenario se caracteriza por tener la mayor densidad poblacional de las tres instancias y una pendiente suave hacia la costa.

En el escenario 2 el algoritmo evolutivo obtuvo una solución con un costo total de 2.581.828,03, frente a la solución obtenida por el algoritmo greedy de 2.960.490,83, lo cual representa una mejora de 12,79 %. La comparación de las métricas físicas de ambas soluciones se presenta en la tabla 5.10, donde se revela que el AE encontró un diseño estructuralmente más eficiente. La diferencia principal radica en la optimización en la excavación, ya que el AE logró una reducción de 16,04 % de volumen de tierra excavada. Esta diferencia se explica visualmente en la figura 5.16, que presenta el perfil de elevación para un camino común entre la solución del algoritmo greedy y del AE, donde el camino construido por el AE se representa en color azul y el del algoritmo greedy en color verde. La solución greedy realizó una caída vertical abrupta en un pozo de sumidero en una etapa temprana del trazado. Esta decisión local lo obliga a mantener el resto de la tubería a una profundidad mayor que la solución del evolutivo. La solución construida por el AE utilizó una ruta más eficiente y mejor adaptada a la topografía del terreno, solo recurre al pozo de

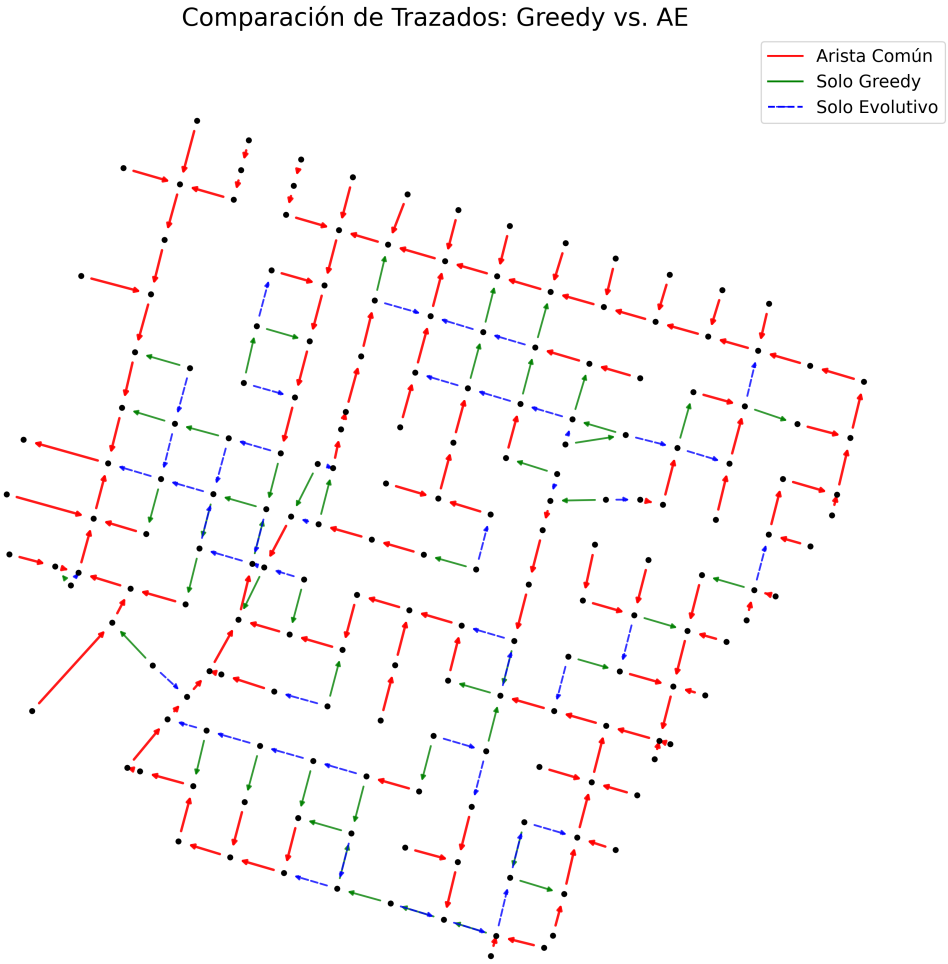


Figura 5.15: Comparación de trazados del escenario 1

sumidero al final del tramo donde es necesario para conectar con las secciones mas bajas de la red.

Métrica	Greedy	AE	% Mejora
Volumen excavación (m^3)	62.049,76	52.098,00	16,04 %
Longitud total tuberías (m)	12.067	10.883	9,81 %
Profundidad máxima (m)	24,28	24,89	-2,51 %
Profundidad promedio (m)	5,14	4,79	6,90 %

Tabla 5.10: Comparativa de las soluciones del algoritmo greedy y el evolutivo para el escenario 2.

En la figura 5.17 se presenta el diseño de la red para ambas soluciones, donde las aristas comunes fueron representadas en color rojo, en verdes las aristas del algoritmo greedy y en azul las del AE. Comparando con el mapa

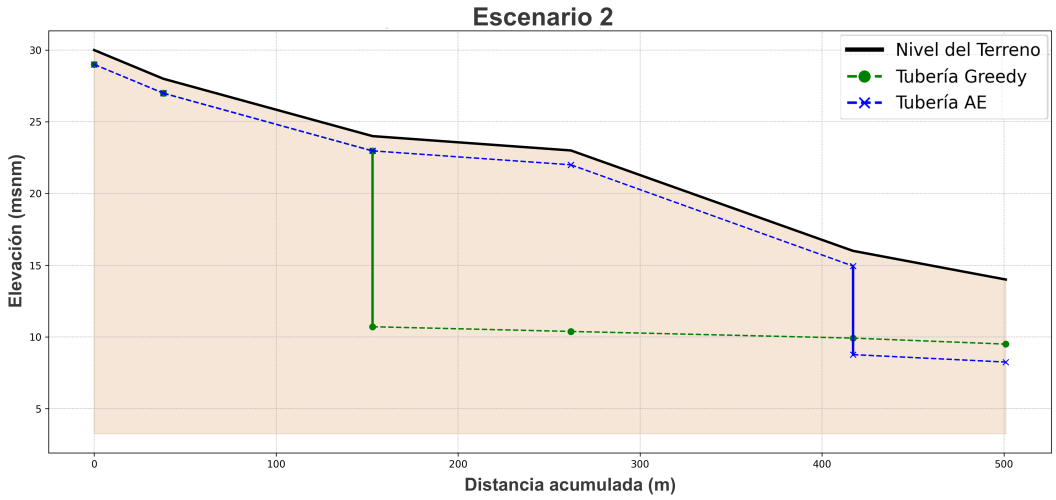


Figura 5.16: Comparación del perfil longitudinal para un camino común en el escenario 2. Se muestra el nivel del terreno (linea negra) y las cotas de las tuberías para la solución greedy (verde) y la del AE (azul).

topográfico del escenario 2 presentado en la figura 5.6 donde la solución construida con el algoritmo greedy tiende a crear conexiones que bajan de forma bastante recta por la pendiente del terreno (desde la zona amarilla/marrón en la parte superior hacia la verde/azul en la inferior). Esto es consistente con la naturaleza del greedy donde en cada punto, elige el camino localmente más óptimo. El AE en cambio, demuestra su capacidad para encontrar soluciones estructuralmente diferentes y menos intuitivas. La diferencia más notable se ve en la zona central del mapa. En lugar de seguir la pendiente de forma directa como el greedy, crea conexiones que atraviesan la pendiente de forma más horizontal antes de continuar el descenso.

Por último en la tabla 5.11 se presenta la distribución de diámetros utilizados en cada solución, dónde se detalla que la solución del AE utilizó menos metros de tuberías de 400 mm que el greedy generando un ahorro en el costo del material.

El escenario 3 corresponde al barrio La Unión, fue diseñado para el desafío particular de un terreno predominantemente llano. Esta particularidad topográfica pone a prueba la capacidad del AE para diseñar una red gravitacional eficiente manteniendo las pendientes requeridas.

Las métricas comparativas de las soluciones presentadas en la tabla 5.12 demostraron que el enfoque evolutivo fue superior al del algoritmo greedy en este tipo de topografías. La mejora más notable se produjo en la profundidad

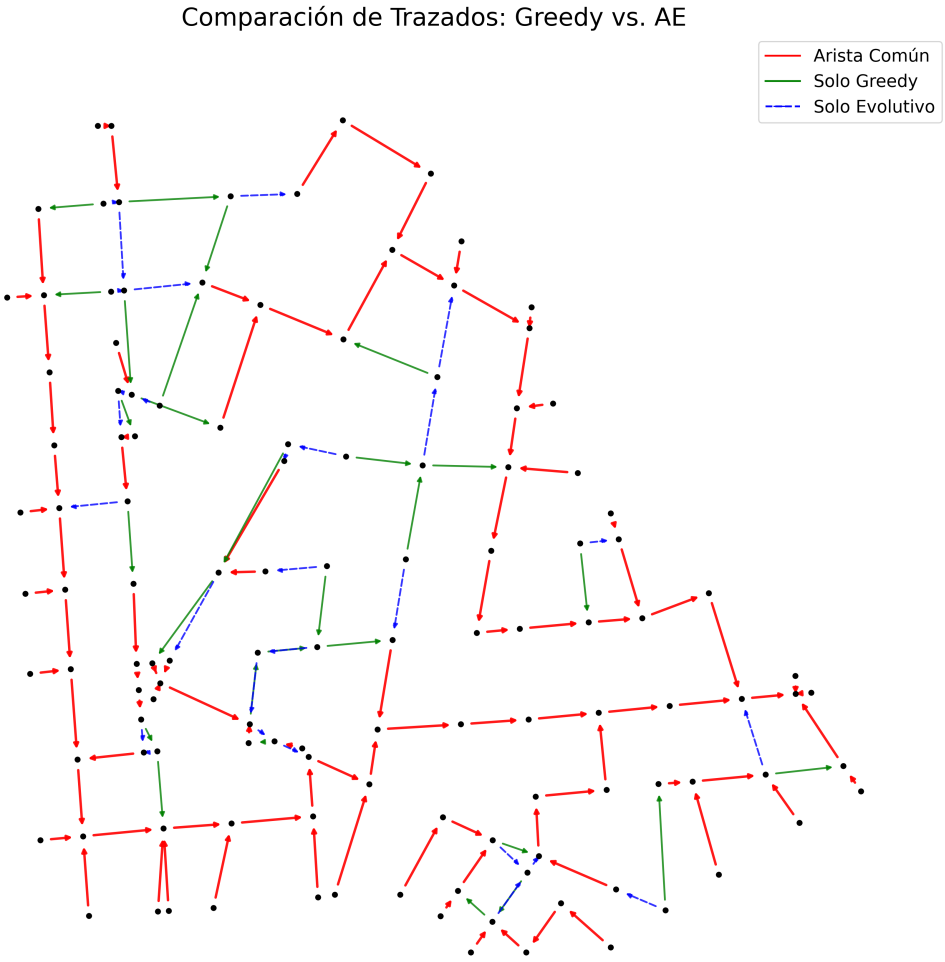


Figura 5.17: Comparación de trazados del escenario 2

Distribución de diámetros (m)	Solución greedy	Solución evolutivo
Longitud tubería 100 mm	7.100	6.580
Longitud tubería 150 mm	1.452	1.428
Longitud tubería 200 mm	687	781
Longitud tubería 250 mm	884	898
Longitud tubería 300 mm	422	528
Longitud tubería 400 mm	1.522	711

Tabla 5.11: Comparativa de la distribución de diámetros utilizados en ambas soluciones en el escenario 2.

máxima alcanzada donde el AE logró una reducción del 21,57 %. A su vez, el AE encontró una solución que excava un 11,31 % menos de volumen total, demostrando su superioridad en la gestión de los recursos.

Métrica	Greedy	AE	% Mejora
Volumen excavación (m^3)	56.081,66	49.739,50	11,31 %
Longitud total tuberías (m)	18.500	17.419	5,84 %
Profundidad máxima (m)	16,08	12,61	21,57 %
Profundidad promedio (m)	3,03	2,86	5,8 %

Tabla 5.12: Comparativa de las soluciones del algoritmo greedy y el evolutivo para el escenario 3.

El análisis visual de los trazados de la red se presenta en la figura 5.18. Los caminos troncales más largos son mayoritariamente rojos, indicando que tanto el greedy como el AE coincidieron en la ruta general de los colectores principales. La optimización realizada por el AE, no radicó en encontrar una topología distinta, sino en realizar ajustes estratégicos a nivel de tramos individuales. Las aristas exclusivas (verdes para el greedy y azules para el AE) no forman caminos largos. En el escenario 3, el AE fue capaz de identificar que un pequeño desvío de uno o dos bloques le permitían aprovechar una topografía más favorable para luego reconectar con la ruta principal. Este desvío local evita una excavación más profunda que el greedy se vio obligado a realizar. La suma de estos múltiples desvíos a lo largo de la red es lo que se traduce en la reducción global del volumen de excavación y la profundidad máxima.

Por último, en la tabla 5.13 se presenta la distribución de diámetros utilizados por cada solución. La solución del AE utilizó significativamente menos metros de tuberías de mayor diámetro, en particular utilizó 375 metros menos de la tuberías mas grande. A su vez, también redujo en 1.006 metros el uso de la tubería de menor diámetro. Dicha reducción se compensó con un mayor uso de las tuberías de diámetro intermedio. Esta estrategia sugiere que el AE logró gestionar el caudal de la red con tuberías de tamaño medio.

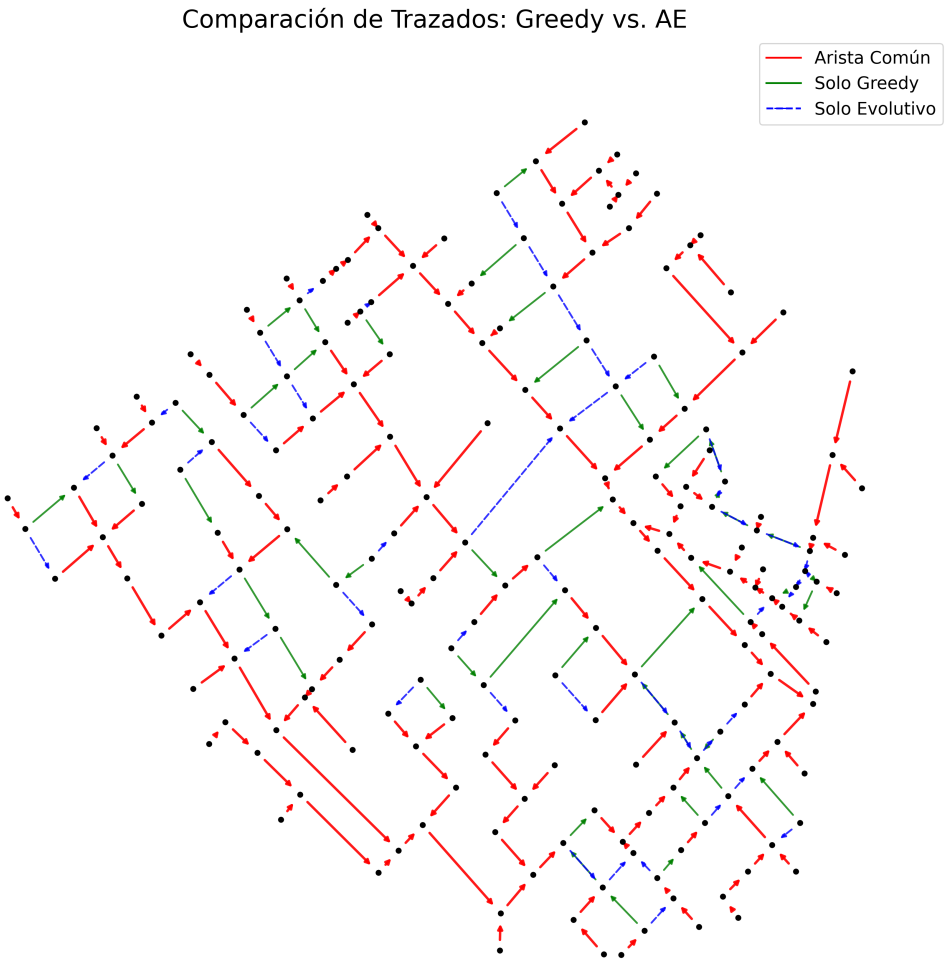


Figura 5.18: Comparación de trazados del escenario 3

Distribución de diámetros (m)	Solución greedy	Solución evolutivo
Longitud tubería 100 mm	10.656	9.650
Longitud tubería 150 mm	2.988	3.404
Longitud tubería 200 mm	1.493	1.430
Longitud tubería 250 mm	1.128	1.046
Longitud tubería 300 mm	781	810
Longitud tubería 400 mm	1.454	1.079

Tabla 5.13: Comparativa de la distribución de diámetros utilizados en ambas soluciones en el escenario 3.

Capítulo 6

Consideraciones finales

Este capítulo presenta las conclusiones del trabajo realizado en este proyecto y las principales líneas de trabajo futuro.

6.1. Conclusiones

En esta tesis se exploró el uso de algoritmos evolutivos como herramienta para resolver el problema del diseño de redes de saneamiento en entornos urbanos, permitiendo encontrar soluciones de buena calidad que cumplen con las restricciones topográficas e hidráulicas del problema. Se partió de la idea de que este tipo de redes debe respetar tanto criterios técnicos como físicos, y al mismo tiempo ser económicamente viables, lo que genera un problema complejo con múltiples restricciones. Para ello, se desarrolló un algoritmo evolutivo capaz de adaptarse a diferentes escenarios reales, utilizando información geográfica y poblacional como base para generar soluciones viables y de bajo costo.

Mediante un análisis experimental exhaustivo, se identificó una configuración óptima del algoritmo que mejoró los resultados obtenidos mediante un enfoque greedy. En los tres escenarios evaluados, el algoritmo evolutivo superó al algoritmo greedy con mejoras del orden del 10 % en el costo total de la red, lo que refuerza la solidez de los resultados obtenidos.

Además del buen desempeño, el algoritmo mostró una variabilidad reducida en sus resultados, lo que evidencia un comportamiento robusto frente a la aleatoriedad del proceso evolutivo. Las gráficas de evolución del fitness también

evidenciaron una tendencia clara de convergencia hacia soluciones estables dentro de un número razonable de generaciones.

Como parte del trabajo, se realizó un relevamiento del estado del arte que permitió conocer los principales enfoques utilizados en el diseño de redes de saneamiento mediante algoritmos evolutivos. Esta revisión sirvió de base para orientar el desarrollo metodológico, identificar buenas prácticas y tomar decisiones informadas sobre la representación del problema, los operadores genéticos y los criterios de evaluación.

6.2. Trabajo futuro

Existen varios aspectos que podrían mejorarse y explorarse en futuras investigaciones. Uno de ellos es la selección del nodo sumidero, que en este proyecto fue definida manualmente mediante un archivo de configuración. Una línea de trabajo interesante sería desarrollar un mecanismo automático para identificar uno o varios nodos sumidero adecuados en función de la topografía y la distribución de la demanda. Esto permitiría una mayor flexibilidad para abordar escenarios más complejos o de mayor escala.

Otro aspecto a considerar es la extensión del enfoque a escenarios urbanos de otras ciudades de Latinoamérica. Si bien el desarrollo y las pruebas se realizaron sobre zonas de Montevideo, la metodología propuesta es aplicable a cualquier ciudad que cuente con datos geográficos y poblacionales adecuados. Esto permitiría evaluar la generalidad del enfoque y adaptarlo a distintas normativas locales.

También se podrían explorar mejoras en la representación de las soluciones, el diseño de nuevos operadores genéticos o la incorporación de técnicas híbridas con heurísticas determinísticas. Asimismo, el modelo de costos utilizado podría enriquecerse incluyendo factores adicionales como costos de mantenimiento, impactos ambientales o consideraciones sociales. Otro punto a mejorar es el manejo de eventos extraordinarios como inundaciones, o prever situaciones donde la demanda cambie debido a un crecimiento en la población de la zona, ya que el enfoque se realizó con la población que existe actualmente.

En resumen, la tesis sienta las bases para seguir explorando el uso de técnicas evolutivas en problemas de infraestructura urbana, particularmente en el contexto latinoamericano, y abre la puerta a múltiples líneas de trabajo que pueden enriquecer y potenciar el enfoque desarrollado. De este modo, se afian-

za el potencial de los algoritmos evolutivos como herramienta para enfrentar desafíos reales de planificación urbana en Latinoamérica.

Bibliografía

Fleischmann, M. (2019). momepy: Urban Morphology Measuring Toolkit. *Journal of Open Source Software*, *4*(43), 1807. https://doi.org/10.21105/joss.01807

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., y Gagné, C. (2012). DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, *13*, 2171-2175.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning* (1st). Addison-Wesley Longman Publishing Co., Inc.

Graph Exchange XML Format. (2022). *Graph Exchange XML Format (GEXF)*. Consultado en marzo de 2022, desde http://gexf.net/

Hagberg, A. A., Schult, D. A., y Swart, P. J. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX. En G. Varoquaux, T. Vaught y J. Millman (Eds.), *Proceedings of the 7th Python in Science Conference* (pp. 11-15).

Haghighi, A., y Bakhshipour, A. E. (2012). Optimization of sewer networks using an adaptive genetic algorithm. *Water resources management*, *26*(12), 3441-3456.

Hassan, W. H., Jassem, M. H., y Mohammed, S. S. (2018). A GA-HP Model for the Optimal Design of Sewer Networks. *Water Resources Management*, *32*, 865-879. https://doi.org/10.1007/s11269-017-1843-y

Hassan, W. H., Attea, Z. H., y Mohammed, S. S. (2020). Optimum layout design of sewer networks by hybrid genetic algorithm. *Journal of Applied Water Engineering and Research*, *8*(2), 108-124. https://doi.org/10.1080/23249676.2020.1761897

Interpolating Point Data. (2024). Consultado en enero de 2024, desde https://www.qgistutorials.com/en/docs/3/interpolating_point_data.html

Jordahl, K., den Bossche, J. V., Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C.,

Hjelle, G. A., Snow, A. D., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., ... Leblanc, F. (2022, diciembre). *geopandas/geopandas: v0.12.2* (Ver. v0.12.2). Zenodo. https://doi.org/10.5281/zenodo.7422493

Liang, L. Y., Thompson, R. G., y Young, D. M. (2000). Designing Wastewater Collection Systems Using Genetic Algorithms. En R. Mizoguchi y J. Slaney (Eds.), *PRICAI 2000 Topics in Artificial Intelligence*. Springer Berlin Heidelberg.

Mann, H. B., y Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, *18*(1), 50-60. https://doi.org/10.1214/aoms/1177730491

Masnsouri, M., y Khanjani, M. (1999). Optimization of Sewer System by Nonlinear Programming. *Journal of Water and Wastewater; Ab va Fazilab (in persian)*, *10*(2), 20-30.

Mays, L. W., y Yen, B. C. (1975). Optimal cost design of branched sewer systems. *Water Resources Research*, *11*(1), 37-47. https://doi.org/https://doi.org/10.1029/WR011i001p00037

Nesmachnow, S., y Iturriaga, S. (2019). Cluster-UY: Collaborative Scientific High Performance Computing in Uruguay. En M. Torres y J. Klapp (Eds.), *Supercomputing* (pp. 188-202). Springer International Publishing.

QGIS Development Team. (2009). *QGIS Geographic Information System*. Open Source Geospatial Foundation. http://qgis.osgeo.org

Rohani, M., y Afshar, M. H. (2015). GA–GHCA model for the optimal design of pumped sewer networks. *Canadian Journal of Civil Engineering*, *42*(1), 1-12.

Weng, H., y Liaw, S. (2015). Establishing an optimization model for sewer system layout with applied genetic algorithm. *Journal of Environmental Informatics*, *5*(1), 26-35.

Zhao, W., Beach, T. H., y Rezgui, Y. (2016). Optimization of Potable Water Distribution and Wastewater Collection Networks: A Systematic Review and Future Research Directions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *46*(5), 659-681. https://doi.org/10.1109/TSMC.2015.2461188