Optimización de Rutas para la Recolección de Residuos Urbanos en Montevideo utilizando Algoritmos Evolutivos

Nicolás Cáceres y Facundo Torterola

***Abstract*—Este informe presenta un modelo de optimización de rutas de recolección de residuos en Montevideo mediante algoritmos evolutivos (AE). Utilizando la geolocalización de contenedores y datos poblacionales, se estima la cantidad inicial de residuos y se optimizan las rutas para minimizar tanto la distancia recorrida como el costo del servicio. El modelo busca reducir el impacto ambiental, minimizando el uso de gasolina y optimizando el costo total del recorrido, promoviendo un servicio de recolección más sustentable.**

***Index Terms*—Algoritmos Evolutivos, Optimización.**

I. Introducción

La recolección de residuos en Montevideo presenta desafíos debido a la dispersión de los contenedores de basura y la alta densidad poblacional en ciertas zonas. Este informe propone optimizar las rutas de recolección mediante AE, los cuales permiten reducir las distancias recorridas, minimizar el consumo de gasolina y asegurar que todos los contenedores sean vaciados en un mismo recorrido. Utilizando un archivo CSV con ubicaciones de los contenedores y datos del censo de 2011, se estima la carga inicial de residuos en cada contenedor.

II. Descripción

El problema de recolección de residuos en Montevideo se enfoca en minimizar el tiempo que cada contenedor permanece lleno de residuos. La intendencia de Montevideo tiene datos públicos [1] en formato CSV que contiene la geolocalización de todos los contenedores de la ciudad, lo que permite establecer rutas que optimicen el desplazamiento de los camiones de recolección.

Con base en el censo de 2011[2], se estima la generación diaria de residuos en cada área, tomando datos de la IM(Intendencia de Montevideo) de cuantos residuos genera una persona[3]. Esta estimación permite asignar una tasa de llenado a cada contenedor, lo cual es fundamental para tener en cuenta la calidad de servicio que se brinda. El objetivo de esta optimación es reducir el costo de la ruta sin perder la calidad de servicio.

III. Justificación del Uso de Algoritmos Evolutivos (AE)

Para el problema de optimización de rutas para la recolección de residuos, los algoritmos evolutivos son una opción altamente adecuada. Este problema, al igual que el Problema del Viajante de Comercio (TSP), pertenece a la clase de problemas NP-difíciles, lo cual implica que no se conoce un algoritmo que garantice una solución óptima en tiempo polinomial para todas las instancias posibles. En nuestro caso, cada camión debe visitar una serie de contenedores distribuidos por la ciudad para recoger los residuos, respetando ciertas limitaciones, como la capacidad de los camiones, y buscando minimizar costos y enfocados en un buen servicio. En nuestro problema, no solo es importante minimizar la distancia total, sino también reducir los costos asociados al consumo de combustible y asegurar un servicio completo (es decir, que todos los contenedores sean vaciados). Los algoritmos evolutivos, como el NSGA-II, permiten optimizar este objetivo, tal como en variantes multiobjetivo del TSP. Esta capacidad de balancear criterios es particularmente útil en la recolección de residuos, donde buscamos un equilibrio entre eficiencia en el servicio y costos ambientales y financieros.

IV. Estrategia de Resolución

Para abordar el problema, se plantea una estrategia de resolución basada en un enfoque de algoritmos evolutivos. El objetivo principal es optimizar la ruta de los camiones de recolección, minimizando la distancia recorrida y, al mismo tiempo, maximizando la calidad del servicio.

A. Objetivos del Problema

El problema se formula con dos objetivos principales:

- **Minimizar la distancia total recorrida (costo de la ruta):** Este objetivo se enfoca en reducir el recorrido total de los camiones para minimizar el consumo de combustible, el tiempo de operación y el impacto ambiental.
- **Maximizar la calidad del servicio (cobertura de recolección):** Este objetivo se mide asegurando que todos los contenedores sean vaciados en cada recorrido. Para ello, la función de fitness penaliza las soluciones en las que los contenedores con alta demanda queden atrás. La calidad del servicio se maximiza reduciendo el tiempo que cada contenedor permanece lleno o cercano a su capacidad máxima, considerando las tasas de generación de residuos de acuerdo con la generación de basura de Monteivdeo.

B. Representación de la Solución

En este problema, cada solución modela la asignación de contenedores a los camiones de recolección. Para garantizar

eficiencia en el uso de memoria, la solución se representa mediante un **arreglo unidimensional** de tamaño n, donde:

- n corresponde al número total de contenedores en la ciudad.
- Cada índice del arreglo representa la posición en el recorrido, indicando el orden en que los contenedores deben ser recogidos.
- El valor almacenado en cada posición del arreglo identifica al contenedor que será recogido en esa posición.

La asignación de camiones a los contenedores se realiza mediante una **partición equitativa**. En este enfoque, el contenedor i es asignado al camión correspondiente según la regla $i \mod$ cantidad_camiones, donde cantidad_camiones es el total de camiones disponibles. Este mecanismo asegura una distribución balanceada de contenedores entre los camiones.

Métrica de éxito: Para garantizar la cobertura completa de recolección, se valida que todos los identificadores de los contenedores estén presentes en el vector de solución, asegurando así que cada contenedor sea asignado a un camión.

- La distribución de carga entre los camiones está garantizada debido a la representación de la solución, que asigna los contenedores de forma equitativa.

Este enfoque no solo asegura que todos los contenedores sean recogidos, sino también que los camiones tengan una carga de trabajo equilibrada, optimizando la eficiencia operativa.

Un ejemplo de vector de solución, suponiendo que hay dos camiones, sería:

$$\text{Solución} = [1, 4, 3, 2]$$

En este caso:

- El **camión** 1 recoge los contenedores 1 y 3.
- El **camión** 2 recoge los contenedores 4 y 2.

Este modelo no solo facilita la validación de la cobertura y la distribución de carga, sino que también permite analizar de manera eficiente el recorrido de cada camión.

C. Función de Fitness

La función de fitness se ha diseñado para optimizar dos objetivos principales: minimizar la distancia total recorrida por los camiones y priorizar la recolección de contenedores con alta demanda en etapas tempranas. Esto permite equilibrar la eficiencia de las rutas y la calidad del servicio ofrecido.

La fórmula utilizada para calcular la función de fitness es:

$$\text{Fitness} = \sum_{i=1}^{\text{cantidadContenedores}} \left(\beta \cdot \text{Distancia}_i - \alpha \cdot \frac{\text{Demanda}_i}{\text{Orden}_i} \right)$$

Donde:

- α: Factor de ponderación para incentivar la recolección de contenedores con mayor demanda en posiciones tempranas. El valor de α se define como $\frac{1}{\text{cantidadCamiones}}$, distribuyendo equitativamente la penalización entre los camiones.
- β: Factor de ponderación para minimizar la distancia recorrida. El valor de β también se define como $\frac{1}{\text{cantidadCamiones}}$, incentivando rutas más cortas de manera equitativa.
- Distancia_i: Distancia total recorrida por el camión para recoger el contenedor i.
- Demanda_i: Demanda normalizada del contenedor recogido en la posición i dentro del recorrido.
- Orden_i: Posición del contenedor dentro de la solución, penalizando asignaciones tardías.

El cálculo de esta función en el algoritmo sigue los siguientes pasos:

1) **Asignación de contenedores:** Cada contenedor es asignado a un camión utilizando la solución propuesta. Los contenedores se procesan en el orden determinado por la permutación dada en la solución.
2) **Cálculo de distancia:** Para cada camión, se calcula la distancia desde su posición actual hasta la posición del contenedor asignado. Esta distancia se acumula en el valor total ponderado por β.
3) **Prioridad de demanda:** Se calcula un puntaje de demanda que da prioridad a contenedores con alta demanda en posiciones tempranas. Este puntaje se suma ponderado por α.
4) **Cálculo final:** La distancia total (Distancia_i) y el puntaje de demanda ($\text{Demanda}_i/\text{Orden}_i$) se combinan en la función de fitness.

El resultado final (Fitness) se establece como el objetivo de optimización de la solución, donde el objetivo es minimizar este valor.

V. Operadores Evolutivos

Para abordar el problema de optimización de rutas, se implementaron operadores evolutivos diseñados para garantizar un equilibrio entre la explotación de soluciones prometedoras y la exploración de nuevas configuraciones en el espacio de búsqueda. Los operadores utilizados fueron:

- **Selección:** Se empleó el método de *torneo binario*, en el que se seleccionan dos soluciones aleatorias de la población, comparando sus valores de fitness. La mejor solución del enfrentamiento se selecciona para participar en la siguiente generación. Este método favorece las soluciones de mayor calidad mientras preserva la diversidad al permitir la selección ocasional de soluciones menos óptimas.
- **Cruce:** Se utilizó el operador de cruce *PMX (Partially Mapped Crossover)*. Este operador permite intercambiar segmentos de los cromosomas entre dos soluciones padres, manteniendo la validez de las asignaciones de contenedores a camiones. Esto facilita la combinación de características eficientes de diferentes soluciones, generando descendientes con rutas potencialmente mejoradas.
- **Mutación:** Para fomentar la diversidad y evitar el estancamiento en óptimos locales, se implementó un operador de mutación de intercambio (*swap mutation*). Este

operador selecciona aleatoriamente dos posiciones en la solución y las intercambia, modificando las asignaciones de contenedores a camiones. La mutación introduce variabilidad, aumentando las probabilidades de explorar configuraciones no visitadas previamente.

VI. Implementación del Algoritmo Evolutivo

El algoritmo empleado para resolver este problema es **NSGA-II**, una metaheurística evolutiva ampliamente utilizada para problemas de optimización multiobjetivo. La implementación del algoritmo incluye los siguientes pasos:

1) **Configuración del Problema:** El problema se modeló mediante la clase `ShortestPathMultCamionesProblem` la cual extiende `AbstractIntegerPermutationProblem`, que define las variables, los límites de las asignaciones y la función objetivo basada en la minimización de la distancia total recorrida y el cumplimiento de la demanda.
2) **Configuración de Operadores:**
   - Operador de cruce: `PMXCrossover`, con una probabilidad de cruce configurada según los parámetros de entrada.
   - Operador de mutación: `PermutationSwapMutation`, con una tasa de mutación ajustable para controlar la variabilidad de las soluciones.
3) **Ejecución del Algoritmo:** El algoritmo se configuró con una población inicial de `poblacion` soluciones y se ejecutó durante `generacion` iteraciones. La clase `AlgorithmRunner` se utilizó para manejar la ejecución del algoritmo.
4) **Selección de la Mejor Solución:** Al finalizar la ejecución, se seleccionó la mejor solución de la población final utilizando un comparador basado en el valor del objetivo, representando la solución más eficiente encontrada.

Estos operadores, en combinación con la estructura del algoritmo NSGA-II, garantizan una búsqueda eficiente en el espacio de soluciones, maximizando la calidad de las rutas y la asignación equilibrada de los contenedores a los camiones.

VII. Datos

Los datos utilizados para modelar el problema fueron obtenidos de dos fuentes principales: el sitio web de la Intendencia de Montevideo (IM) y el Instituto Nacional de Estadística (INE). Desde el portal de la IM, se recopiló información sobre la ubicación geográfica de los contenedores dentro de la ciudad, su capacidad de almacenamiento, la densidad promedio de los residuos y el tamaño de la flota de vehículos disponibles. La capacidad de los vehículos recolectores fue obtenida a partir de las especificaciones proporcionadas por los fabricantes.

Por otra parte, los datos sobre la población por manzana se obtuvieron del censo de 2011 provisto por el INE [2]. Como estos datos estaban identificados mediante un código interno (Sección, Segmento y Zona censal), fue necesario cruzarlos con un mapa vectorial también suministrado por el INE. Este procedimiento permitió representar cada manzana mediante las coordenadas de su centroide.

La demanda de los contenedores se calculó en función de la población cercana a cada uno. Para ello, se recorrió cada manzana y se distribuyó equitativamente su población entre los contenedores ubicados a una distancia menor a 200 metros.

Debido a la gran cantidad de contenedores disponibles en Montevideo (11,645 en total), se decidió reducir el alcance del problema y enfocar el análisis únicamente en los municipios B, C y CH, los cuales concentran un total de 4,185 contenedores.

VIII. Evaluación

Según los datos proporcionados por la Intendencia, se identificó la disponibilidad de 27 camiones para llevar a cabo las rutas de recolección en dichos municipios. El modelo del problema sé instancia utilizando tres parámetros principales:

- **Contenedores**: Conjunto de contenedores que serán recogidos durante el recorrido.
- **Camiones**: Vehículos de recolección, cada uno con un contenedor inicial asignado.
- **Probabilidad Greedy**: Probabilidad configurada que determina la toma de decisiones mediante un enfoque ”greedy”. Este enfoque permite que, en cada paso, el camión decida cuál contenedor visitar a continuación, priorizando aquellos que se encuentren más próximos o con mayor demanda, según su ubicación actual.

A. Selección del Mejor Contenedor Greedy

```
private Contenedor
    obtenerMejorContenedorGreedy(Camion camion
    , List<Contenedor> contenedoresSinVisitar)
     {
    return contenedoresSinVisitar.stream()
            .min(Comparator.comparingDouble(c
                -> {
                if (camion.
                    getCapacidadUtilizada() +
                    c.getDemanda() > camion.
                    getCapacidad()) {
                    return Double.MAX_VALUE;
                }
                double distance = camion.
                    getPosicionActual().
                    calcularDistancia(c.
                    getPosicion());
                return distance - (pesoBasura
                    * c.getDemandaNormalizada
                    ());
            }))
}
```

Listing 1. Método para seleccionar el mejor contenedor en un enfoque greedy.

1) Explicación del Método: El método `obtenerMejorContenedorGreedy` realiza las siguientes operaciones:

- **Filtrado por capacidad:** Antes de calcular la distancia, verifica si agregar el contenedor seleccionado excedería la

capacidad del camión. Si es así, asigna un valor máximo (`Double.MAX_VALUE`) para evitar seleccionarlo.
- **Cálculo de distancia:** Calcula la distancia entre la posición actual del camión y el contenedor utilizando el método `calcularDistancia`.
- **Demanda normalizada:** Penaliza el valor de distancia considerando la demanda normalizada del contenedor, ponderada por un factor de peso denominado `pesoBasura` que es α.
- **Selección del mejor contenedor:** Utiliza el método `min()` para encontrar el contenedor con el valor más bajo según los criterios definidos.

IX. Evaluación

El modelo se configuró con los siguientes parámetros para el algoritmo genético:

- **Población inicial**: Se configuraron tamaños de población inicial [50, 100, 200, 500] individuos, explorando diferentes valores para evaluar la diversidad y la calidad de las soluciones.
- **Máximo de generaciones**: Se establecieron límites entre [50,100,200, 500] generaciones, para evaluar la convergencia del algoritmo.
- **Probabilidad de cruce**: Se probaron valores entre [0.7, 0.8, 0.9, 1.0], configurando el operador de cruce PMX (*Partially Mapped Crossover*).
- **Probabilidad de mutación**: Se configuraron valores entre[0.01, 0.05, 0.1, 0.2], utilizando un operador de mutación por intercambio.
- **Probabilidad Greedy**: Se estableció una probabilidad entre [0.1, 0.3, 0.5, 0.9] para controlar el enfoque ”greedy” en la toma de decisiones del camión durante la construcción de las rutas.

Finalmente, los resultados de las configuraciones evaluadas se almacenaron en un archivo CSV, para análisis posterior, cabe destacar que se realizaron 8 ejecuciones de lo anterior mencionado[4]. Este archivo incluye las combinaciones de parámetros utilizadas, la distancia total recorrida por los camiones y las rutas específicas generadas para cada solución, ademas de el tiempo de ejecucion que tardo el algoritmo. Además, se identificó y reportó la mejor solución global entre todas las combinaciones de parámetros probadas.

X. Resultados y Evaluación

A. Relación entre Parámetros y Fitness

1) Tamaño de la Población vs Fitness: El análisis muestra que un tamaño de población mayor tiende a mejorar el valor promedio del fitness (más negativo), lo que indica que poblaciones más grandes permiten una mejor exploración del espacio de soluciones. Los resultados obtenidos son los siguientes:

Tamaño de Población	Fitness Promedio
50	-46.30
100	-47.65
200	-49.29
500	-51.37

TABLE I
Relación entre Tamaño de la Población y Fitness.

2) Número de Generaciones vs Fitness: Se observa que incrementar el número de generaciones correlaciona con una mejora en el fitness promedio, aunque los beneficios tienden a estabilizarse en configuraciones con valores elevados. Los resultados obtenidos son:

Número de Generaciones	Fitness Promedio
50	-47.67
100	-48.25
200	-48.79
500	-49.99

TABLE II
Relación entre Número de Generaciones y Fitness.

3) Probabilidad de Mutación vs Fitness: Existe un punto óptimo para la probabilidad de mutación, con valores intermedios (0.10–0.20) siendo los más efectivos. Los resultados se presentan en la Tabla III.

Probabilidad de Mutación	Fitness Promedio
0.01	-48.04
0.05	-48.44
0.10	-48.86
0.20	-49.37

TABLE III
Relación entre Probabilidad de Mutación y Fitness.

4) Greedy o Mutacion: Como se ve en la Fig. 1, se logran buenos valores de fitness para valores altos de probabilidad de greedy y probabilidad de mutación.

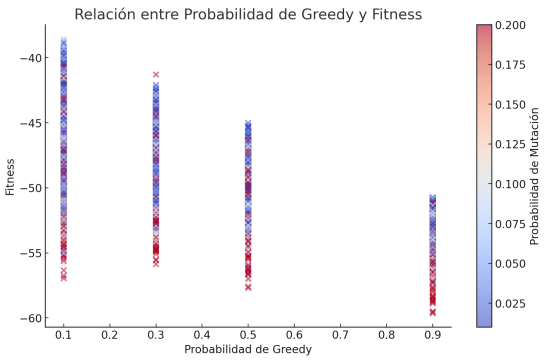

Fig. 1. Relación entre probabilidad de greedy y fitness.

En las Fig. 2 y Fig. 3 vemos como se comporta el tiempo de ejecución promedio al aumentar la probabilidad de greedy así como la probabilidad de mutación. Se puede observar que aumentar al probabilidad de greedy tiene un gran costo computacional, ya que aumenta en gran medida el tiempo de ejecución, mientras que modificar la probabilidad de mutación no genera un impacto significativo al tiempo de ejecución.

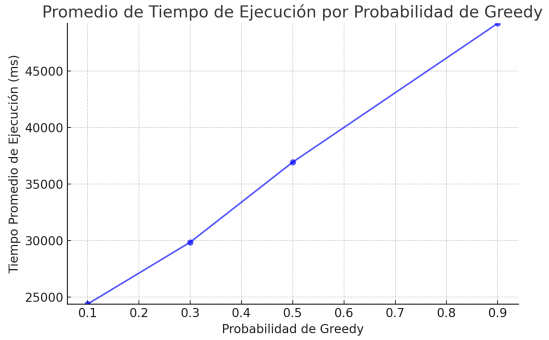

Fig. 2. Relación entre probabilidad de greedy y tiempo de ejecución.

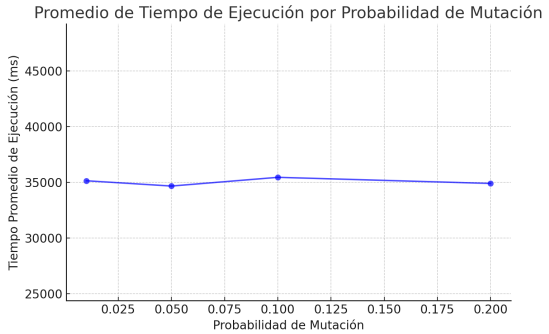

Fig. 3. Relación entre probabilidad de mutación y tiempo de ejecución.

B. Análisis de Tiempos de Ejecución

1) Tiempos Promedio y Extremos: El análisis de tiempos de ejecución arroja las siguientes estadísticas:

- **Tiempo promedio:** 35,028 ms (˜35 segundos).
- **Tiempo mínimo:** 1,275 ms (˜1.3 segundos).
- **Tiempo máximo:** 1,107,861 ms (˜18.5 minutos).
- **Mediana:** 17,169 ms (˜17.2 segundos).

2) Relación entre Tamaño de Población y Generaciones: El tiempo de ejecución aumenta con el tamaño de la población y el número de generaciones, como se observa en la Figura 4.

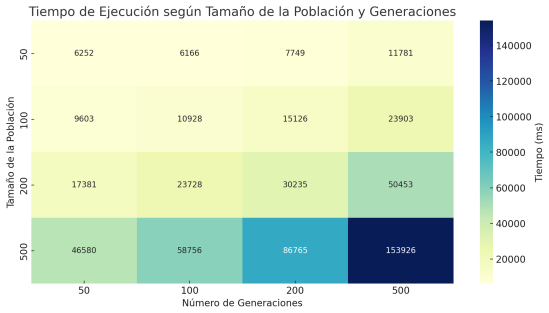

Fig. 4. Tiempo de Ejecución según Tamaño de la Población y Generaciones.

3) Relación entre Fitness y Tiempo de Ejecución: Las soluciones con mejores valores de fitness tienden a requerir más tiempo de ejecución, reflejando el costo computacional de explorar a fondo el espacio de soluciones.

C. Mejor Solución

La mejor solución encontrada tiene las siguientes configuraciones:

- **Tamaño de la Población:** 500.
- **Número de Generaciones:** 500.
- **Probabilidad de Greedy:** 0.9.
- **Probabilidad de Cruce:** 0.8.
- **Probabilidad de Mutación:** 0.2.
- **Fitness:** -59.64.
- **Tiempo de Ejecución:** 193,139 ms (˜193 segundos).

D. Estadísticas Generales

Las estadísticas generales de los parámetros se presentan en la Tabla IV.

Parámetro	Media	Mínima	Máxima
Población	214	50	500
Generaciones	213	50	500
Greedy	0.45	0.1	0.9
Cruce	0.85	0.7	1.0
Mutación	0.09	0.01	0.2
Fitness	-48.67	-59.64	-38.51

TABLE IV
ESTADÍSTICAS GENERALES DE LOS PARÁMETROS.

E. Comparación de Resultados: Greedy vs Algoritmo Evolutivo

Al evaluar los resultados obtenidos utilizando una solución 100% greedy, se obtuvo un valor de fitness de -51.22. Este resultado es significativamente inferior al de nuestra mejor solución (-59.64), obtenida mediante el algoritmo evolutivo.

Esto evidencia que una estrategia puramente greedy, al priorizar decisiones locales óptimas en cada paso, tiende a quedarse atrapada en mínimos locales, limitando su capacidad para encontrar soluciones globalmente óptimas. Por el contrario, el enfoque del algoritmo evolutivo, al combinar exploración y explotación mediante técnicas como el cruce, la mutación y el uso de múltiples generaciones, permite escapar de estos mínimos locales y alcanzar soluciones más eficientes.

XI. Conclusiones y Trabajo Futuro

A. Conclusiones

El uso de algoritmos evolutivos en la optimización de rutas para la recolección de residuos ha demostrado ser efectivo para equilibrar la distancia recorrida y la calidad del servicio. A continuación, se resumen las conclusiones clave:

- **Eficiencia en la Solución Global:** Los algoritmos evolutivos superaron significativamente a estrategias puramente greedy, evitando mínimos locales y encontrando soluciones globalmente óptimas. La mejor solución obtenida alcanzó un fitness de -59.64, frente al -51.22 del enfoque greedy.
- **Impacto de los Parámetros:** Configuraciones con alta población y múltiples generaciones lograron mejores resultados, aunque a un mayor costo computacional. Las probabilidades de mutación intermedias (0.1–0.2)

ofrecieron un equilibrio óptimo entre exploración y explotación.
- **Costo Computacional:** Si bien los tiempos de ejecución fueron elevados, los resultados obtenidos justifican el esfuerzo computacional en problemas donde la calidad de la solución es prioritaria.
- **Factores Externos:** La ubicación del depósito de residuos se identificó como un factor crítico para la eficiencia del recorrido. La implementación de múltiples puntos estratégicos de despacho de vehículos podría mejorar notablemente los resultados.
- **Monitoreo en Tiempo Real:** La incorporación de métricas en tiempo real sobre el nivel de llenado de los contenedores permitiría una planificación más dinámica y eficiente, adaptándose a las necesidades del servicio.

B. Trabajo Futuro

Con base en los resultados obtenidos, se proponen las siguientes líneas de investigación y desarrollo:

1) **Mejoras en el Modelo:**
   - Incorporar restricciones adicionales, como ventanas temporales o limitaciones específicas de capacidad, para hacer el modelo más realista.
   - Experimentar con otras variantes de algoritmos evolutivos, como el NSGA-III o enfoques híbridos que combinen heurísticas y metaheurísticas.
2) **Optimización Computacional:**
   - Implementar técnicas de paralelización para reducir los tiempos de ejecución en configuraciones con alta población y generaciones.
   - Explorar el uso de GPUs para acelerar la evaluación de soluciones, mejorando significativamente la eficiencia del algoritmo.
3) **Validación en Escenarios Reales:**
   - Aplicar el modelo a datos reales y comparar su desempeño con los métodos actualmente implementados en el sistema de recolección de residuos.
   - Realizar simulaciones basadas en datos históricos para evaluar la robustez y escalabilidad de las soluciones propuestas.
4) **Extensión Multiobjetivo:**
   - Reformular el problema como un modelo multiobjetivo que incluya objetivos adicionales, maximizar la satisfacción del usuario final.
5) **Integración con Datos en Tiempo Real:**
   - Incorporar sensores IoT para medir el llenado de los contenedores en tiempo real, permitiendo rutas dinámicas y priorización de áreas críticas.

Estas propuestas tienen el potencial de mejorar significativamente la eficiencia, sostenibilidad y adaptabilidad del sistema de recolección de residuos en entornos urbanos.

REFERENCES

[1] (2024) Datos contenedores domiciliarios. [Online]. Available: https://ckan-data.montevideo.gub.uy/vistas/levantes-contenedores?format=csv

[2] (2024) Datos del censo de montevideo. [Online]. Available: https://www.gub.uy/instituto-nacional-estadistica/datos-y-estadisticas/estadisticas/marcos-censales

[3] (2021) Datos sobre la cantidad de basura producida por una persona. [Online]. Available: https://montevideo.gub.uy/sites/default/files/biblioteca/20210823criteriosdeubicacioncontenedo

[4] Repositorio de git. [Online]. Available: https://github.com/facundotorterola/ae-grupo-p-2024/tree/main