

# Algoritmos Evolutivos - 2024

## Informe Inicial

Santiago Gestal

Facultad de Ingeniería, UDELAR  
Montevideo, Uruguay  
[santiago.gestal@fing.edu.uy](mailto:santiago.gestal@fing.edu.uy)

Santiago Lopez de Haro

Facultad de Ingeniería, UDELAR  
Montevideo, Uruguay  
[santiago.haro.grille@fing.edu.uy](mailto:santiago.haro.grille@fing.edu.uy)

**Resumen**—En este informe se presenta una implementación de un algoritmo evolutivo que busca optimizar la ubicación de paradas de autobuses en una ciudad considerando costos de instalación y calidad de servicio, en particular la cobertura de la demanda y el tiempo de viaje.

### I. DESCRIPCIÓN DEL PROBLEMA

Un sistema de transporte público optimizado representa un aspecto fundamental para el desarrollo y sostenibilidad de una ciudad. Su relevancia se extiende a diversas áreas como podrían ser la disminución del tráfico, impacto ambiental, accesibilidad, calidad de vida, entre otros.

En consideración de lo mencionado anteriormente, se eligió para este proyecto centrarse en la presentación de un algoritmo evolutivo capaz de optimizar la ubicación de las paradas de autobuses en una ciudad, de tal manera que se maximice la eficiencia del sistema de transporte público. Para esto se tomarán en cuenta los siguientes criterios:

- Económicos: en particular, el costo asociado a la instalación de dichas paradas de acuerdo a el tipo de las mismas (sin refugio, con refugio simple, con refugio doble).
- De calidad de servicio: como podrían ser una cobertura adecuada de la demanda y minimización del tiempo de viaje.

Para modelar la demanda, se utilizará el concepto de matriz origen-destino, la cual permite visualizar los patrones de desplazamiento de las personas al exhibir la cantidad de viajes realizados entre un cierto lugar de origen y otro de destino para cada par de ubicaciones en una red de transporte.

La división considerada para discriminar las ubicaciones en una ciudad será la definida por los segmentos censales de la misma.

El algoritmo buscará determinar en que segmentos censales conviene colocar qué tipo de parada, si es que alguna, y después con la solución dada de forma ajena al algoritmo se podrá tomar la decisión de cuántas y en que posición exacta colocar cada parada.

Estos datos fueron extraídos a partir de una investigación realizada como proyecto de tesis, en la cual se hizo un análisis de la movilidad urbana en Montevideo. [1] Además, como instancia complementaria se obtuvo de un repositorio de github los datos de movilidad de Buenos Aires. [2]

### II. JUSTIFICACIÓN DE USAR AE

El problema presentado anteriormente cuenta con una naturaleza compleja y multiobjetivo. A su vez, el número de combinaciones de ubicaciones de paradas es muy elevado, lo que hace que una búsqueda exhaustiva sea computacionalmente inviable.

Tomando esto en cuenta, el uso de un algoritmo evolutivo parece ser adecuado para la resolución de este problema. Esto se debe a su capacidad de explorar grandes espacios de búsqueda para hallar soluciones aptas en tiempos de ejecución razonables. Además, los mismos nos permiten manejar problemas con múltiples objetivos de manera eficiente usando, por ejemplo, la técnica de frente de Pareto, hallando soluciones que representen compromisos entre las diferentes metas sin necesidad de formular una función objetivo única.

### III. ESTRATEGIA DE RESOLUCIÓN

#### A. Representación de individuos

Para la representación de los individuos usaremos una tira de números enteros, en el rango de 0 a 3, dónde 0 representa que en esa zona no conviene colocar paradas, y el rango de 1 a 3 representa que tipo de parada conviene colocar. Cuanto mayor sea el número determinado para el segmento mayor es la calidad de la parada. Se buscará que en las zonas con mayor demanda se incluyan paradas de mayor calidad. Luego, con algún otro mecanismo se decide la cantidad dentro de la zona y la posición de las mismas, pero esa parte del problema es ajeno a la solución dada por el algoritmo.

El largo de la tira será igual al de la cantidad de segmentos de censo en la instancia naturalmente.

## B. Formulación matemática

Para esta sección se considerarán los siguientes tipos de parada:

$$T = \{sin\_refugio, refugio\_simple, refugio\_doble\}$$

Donde el costo de cada una de ellas será igual a su número correspondiente. Es decir:

- $\text{costo}(sin\_refugio) = 1$
- $\text{costo}(refugio\_simple) = 2$
- $\text{costo}(refugio\_doble) = 3$

Además, se considerará  $\text{costo}(sin\_parada) = 0$

Una entrada de la matriz origen-destino está denotada como  $OD_{i,j}$  donde  $i$  representa el segmento censal de origen, y  $j$  el de destino, y el valor es la cantidad de personas que viajan de  $i$  a  $j$  en un cierto intervalo de tiempo. Se asume que la matriz tiene un tamaño de  $m * n$ .

Finalmente, se define la  $\text{demanda}(i), i \in \text{Segmentos}$  como:

$$\text{demanda}(i) = \sum_k OD_{i,k} + \sum_r OD_{r,i}$$

Siendo  $k$  y  $r$  todos los segmentos en la matriz OD para los que  $i$  aparece como origen y destino respectivamente.

## C. Funciones objetivo

Luego, dado un individuo  $x$ , se quiere maximizar  $F(x) = (f_1(x), f_2(x), f_3(x))$ , donde las componentes que se buscan optimizar son:

- 1) Maximización de cobertura: para cada segmento censal  $i$ , el número de pasajeros  $OD_{i,j}$  que necesitan llegar a otro segmento  $j$  debe tener acceso a una parada. De igual forma debe existir una parada en  $j$  para que la persona pueda descender del autobús.

$$f_1(x) = \sum_{i=1}^m \sum_{j=1}^n (OD_{i,j} + OD_{j,i}) * \text{cubierto}(i, j, x)$$

Donde:

$$\text{cubierto}(i, j, x) =$$

$$\begin{cases} 1 & \text{si los segmentos } i \text{ y } j \text{ están cubiertos en } x \\ 0 & \text{si no} \end{cases}$$

- 2) Minimización de tiempo de viaje: asumiendo que al reducir la cantidad de paradas el transporte tendrá que parar menos, se tarda menos tiempo en completar el recorrido. (Notar que se multiplica por -1 dado que el formalismo de AE propone maximizar el fitness)

$$f_2(x) = (-1) \sum_{i \in T} \text{cantidad}_i(x)$$

- 3) Minimización del costo tomando en cuenta la demanda que tiene el segmento (nuevamente, multiplicamos por -1):

$$f_3(x) = (-1) \sum_{i \in \text{Segmentos}} \text{costo}_i(x) * D_i$$

$$\text{Con } D_i = 2 * (1 - \frac{\text{demanda}(i)}{\max(\text{demanda})}) - 1$$

El coeficiente  $D_i$  oscila entre -1 y 1 dependiendo de la demanda que posea ese segmento, haciendo que los segmentos con mayor demanda tiendan a tener paradas más caras debido a que el costo va a ser negativo, mientras que los segmentos con menos demanda tenderán a tener paradas más económicas debido a que su costo será positivo.

## D. Operadores

1) *Selección*: Para la selección de los individuos de la población usaremos el usado típicamente en metodologías como NSGA-II, más concretamente usaremos el BinaryTournamentSelection ya que ha demostrado dar buenos resultados para problemas multi-objetivo.

2) *Cruzamiento*: Para la parte de cruzamiento usaremos el método de cruzamiento de 2 puntos, dado el gran tamaño de nuestros individuos.

3) *Mutación*: Para la mutación usaremos el operador IntegerPolynomialMutation, que cada número de la tira tendrá una probabilidad dada de cambiar su valor a otro dentro del rango de forma aleatoria también, la probabilidad será determinada nuevamente con pruebas experimentales pero rondará entre 0.1, 0.01 y 0.001.

4) *Técnicas avanzadas*: Dado que nos encontramos trabajando en un problema multiobjetivo, y utilizaremos un modelo basado en pareto para la resolución del mismo, resulta indispensable incluir un mecanismo que permita mantener la diversidad en la población para poder muestrear el frente de pareto adecuadamente. En particular, dado que nos estamos basando en NSGA-II, utilizaremos el mecanismo de crowding que viene implementado en este.

#### IV. PROPUESTA DE EVALUACIÓN EXPERIMENTAL

1) *Generación de instancias:* Para las instancias de la evaluación usaremos una serie de matrices origen destino de 2 ciudades, Montevideo y Buenos Aires.

En la matriz origen-destino se simula la demanda para el transporte público señalando que cantidad de personas tiene como origen cierto segmento y destino cierto otro.

2) *Comparación con otras técnicas:* Para la comparación con otras técnicas implementaremos un algoritmo de tipo greedy para encontrar una solución optima, con el fin de evaluar la diferencia en tiempo de cómputo y en la calidad de la solución resultante.

3) *Calidad de soluciones:* Como métrica en la calidad de la solución simplemente se usará la función de fitness ya definida para evaluar cual de las dos da mejores resultados.

4) *Eficiencia computacional:* Para evaluar la eficiencia usaremos el tiempo de ejecución promedio de las ejecuciones realizadas en segundos y su desviación estándar.

El equipo usado sera una Macbook Pro 2020 con un Apple M1 y 16 gb LPDDR4 de ram usando como sistema operativo Mac OS 15.1

#### V. DECISIONES DE IMPLEMENTACIÓN

Para la implementación del algoritmo se usó Java 17 junto al framework de JMetal. El problema se modeló como un "IntegerProblem", con operador de cruzamiento "TwoPointCrossover", operador de mutacion "IntegerPolynomialMutation" y como operador de selección el "BinaryTournamentSelection" con el comparador "RankingAndCrowdingDistanceComparator".

Para el algoritmo y su builder creamos clases propias que llamamos "CustomAlgorithm" y "CustomAlgorithmBuilder", basados en la clase de JMetal NSGA-II. Se introdujeron modificaciones como la integracion de un fitness tracker, una clase que creamos llamada "FitnessTracker" que se encarga de llevar y al finalizar guardar en un csv la evolucion del fitness del algoritmo.

Para el modelado del problema se definió una clase llamada "ParadasProblem" extiendiendo la clase "AbstractIntegerProblem". El constructor recibe un Map como parametro que representa la instancia del problema, es decir los segmentos y sus demandas, y en base a esto define la cantidad de variables que serán evaluadas, define que posición de los genes van a representar a que segmento y establece que debe ser un número entre 0 y 3.

Con el fin de optimizar lo más posible la función de fitness se establecieron una serie de estructuras para bajar el tiempo

computacional de acceso a ciertos datos. Por ejemplo, se creó un Map que contiene la demanda total de cada segmento, de forma que no hay que recalcularlo en cada iteracion. A su vez se tiene tanto una estructura para mapear índices a segmentos como para mapear segmentos a índices.

Dado que al intentar ejecutarlo de forma paralela se terminaron teniendo peores tiempos de ejecución, y considerando el overhead de manejar los hilos y que la función de fitness es relativamente sencilla, se decidió ejecutar el algoritmo en un solo núcleo.

#### VI. EVALUACIÓN DE LOS RESULTADOS

##### A. Configuración paramétrica

Con el objetivo de poder realizar la configuración paramétrica, se buscó una instancia del problema de tamaño menor a las mencionadas anteriormente, para poder ejecutar todas las combinaciones posibles en un tiempo razonable. En particular, se utilizó una matriz OD de la ciudad de Pittsburgh, Estados Unidos. [3]

Se seleccionaron los siguientes rangos candidatos:

- Tamaño de población: 50, 100, 200
- Probabilidad de cruce: 0.6, 0.7, 0.8
- Probabilidad de mutación: 0.1, 0.01, 0.001

Y se analizaron las 27 combinaciones posibles entre ellos tomando un total de 30 ejecuciones para cada combinación. Se utilizó para compararlas entre sí la métrica de hipervolumen como referencia de calidad y diversidad de la solución. Además, se registraron el tiempo promedio para cada combinación. (Ver Anexos: Tabla I)

A estos resultados de hipervolumen para cada configuración se les realizó un test de normalidad Kolmogorov-Smirnov, hallando que en la gran mayoría de los casos se obtuvieron valores de p-valor mayores a 0.05, por lo que no se rechaza la hipótesis de que los valores sigan una distribución normal. Es por esto que se realizaron tests paramétricos para el análisis de los resultados.

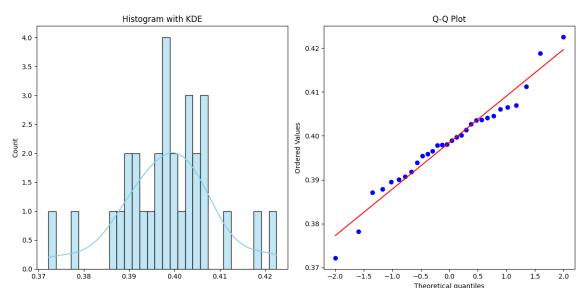


Fig. 1. Ejemplo de histograma con KDE y QQ plot resultante

Dado que contamos con 27 configuraciones, se realizó un test ANOVA (Analysis of variance) para analizar la media de las mismas y determinar si existen diferencias significativas entre ellas. Como era de esperarse de los datos obtenidos en la tabla, se obtuvo un p-valor bajo, lo que indica que se rechaza la hipótesis nula, implicando una diferencia significante entre las diferentes medias.

Luego, se realizó el siguiente boxplot, el cual nos permite representar gráficamente los valores de las configuraciones. La línea central en cada caja representa la media, los extremos izquierdo y derecho de las mismas indican los cuartiles 25 y 75 respectivamente. Las líneas que se extienden desde cada caja se llaman bigotes, y representan la varianza esperada de los datos. Si hay datos que quedan más allá de los extremos de los bigotes, se representan con puntos conocidos como valores atípicos:

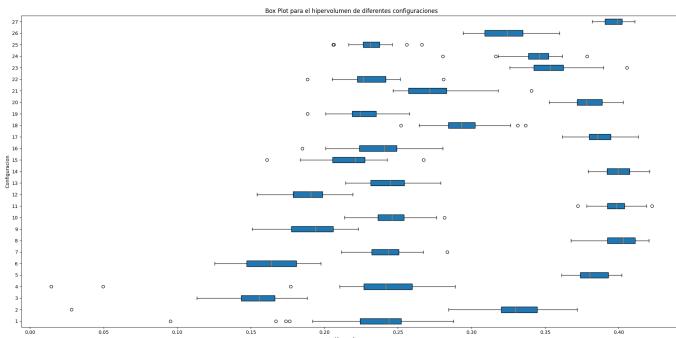


Fig. 2. Box plot para las configuraciones paramétricas

De esto se deduce que la configuración 27 fue la que presentó uno de los valores más altos de media, y con una rango intercuartil pequeño. Esto significa que consiguió valores más altos que las demás configuraciones de manera consistente. Luego, se decidió trabajar con una población de 200 individuos, con probabilidad de cruzamiento 0.8 y probabilidad de mutación 0.1. Cabe mencionar que tiene sentido tener una probabilidad de mutación alta dado el gran tamaño de los individuos, lo que nos permite muestrear el frente de pareto de forma más diversa.

### B. Evaluación experimental

Como se mencionó anteriormente, utilizaremos el algoritmo para determinar en qué segmentos conviene más poner paradas y de qué calidad deben ser las mismas. Para esta etapa de evaluación, se hicieron 30 ejecuciones del algoritmo en Montevideo, y, dado el gran tamaño de la instancia de Buenos Aires, 20 ejecuciones en ese caso.

1) Montevideo: en primer lugar y con el objetivo de poder realizar las validaciones correspondientes, se construyó un frente de pareto aproximado con los resultados de las 30 ejecuciones. Esto se realiza combinando todas las soluciones no

dominadas halladas. Dado que la función objetivo cuenta con tres componentes, el gráfico se encuentra en tres dimensiones. El mismo se presenta a continuación:

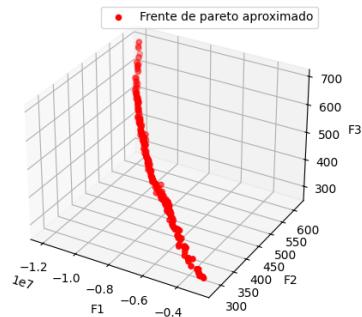


Fig. 3. Frente de pareto aproximado (Mvd)

Para mejor visualización, se muestran las distintas combinaciones de las componentes proyectadas sobre el plano:

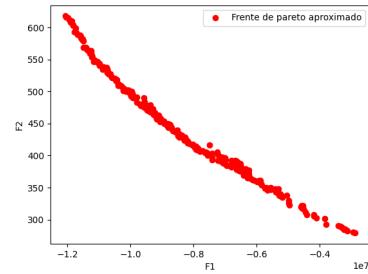


Fig. 4. Proyección en f1 y f2 del frente aproximado (Mvd)

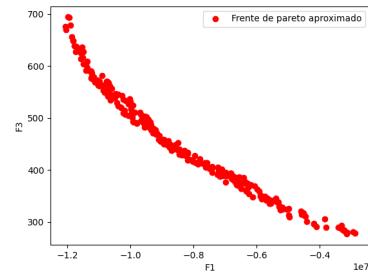


Fig. 5. Proyección en f1 y f3 del frente aproximado (Mvd)

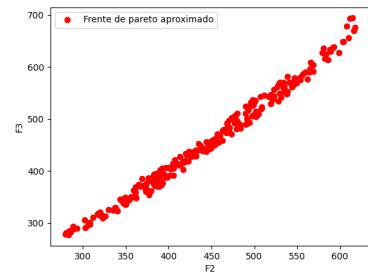


Fig. 6. Proyección en f2 y f3 del frente aproximado (Mvd)

Una vez hecho esto, podemos comparar los resultados de las 30 ejecuciones con el frente de pareto aproximado para reportar los valores de las métricas deseadas. En nuestro caso, utilizamos las funcionalidades implementadas en jMetal para evaluar la tasa de error, distancia generacional, spread, e hipervolumen. (Ver Anexos: Tabla II)

A modo de ejemplo se graficaron las comparaciones entre los frentes de pareto de dos ejecuciones diferentes y el frente de pareto aproximado, siendo estas las ejecuciones que obtuvieron el mejor y peor valor de spread respectivamente (Ver Anexos, fig. 15 y fig 16)

Luego, calculamos el mejor valor, promedio y desviación estándar de cada métrica, así como nuevamente un test de Kolmogorov-Smirnov para ver si los resultados obtenidos siguen una distribución normal. Se extrajeron los siguientes datos:

METRICA	PROM	DESV_EST	MEJOR	P-VALOR
Tasa_error	0.9444	0.1303	0.4297	0.0002
Distancia_generacional	0.0039	0.0012	0.0012	0.584
Spread	0.5347	0.0589	0.4001	0.7284
Hipervolumen	0.4264	0.0589	0.5691	0.4636

Observamos que las métricas de distancia generacional, spread e hipervolumen tienen un p-valor alto, por lo que no se puede rechazar la hipótesis nula con certeza. Por otro lado, la tasa de error tiene un p-valor bajo, indicando que los valores no se corresponden con una distribución normal.

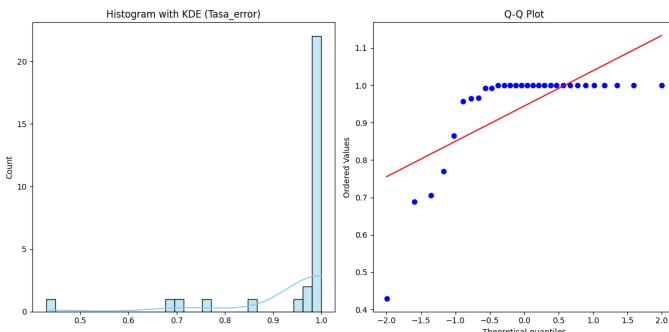


Fig. 7. Histograma con KDE y QQ plot para la tasa de error

Ahora observaremos cómo el fitness va evolucionando en cada generación, para esto veremos por separado el comportamiento de cada uno de los objetivos.

Como se puede apreciar en las siguientes gráficas de las figuras 8 a 10, que muestran la evolución en el caso de Montevideo, el algoritmo mejora el valor de los objetivos hasta que finalmente en las últimas generaciones el valor se estanca, llegando a una solución final que se mantiene por varias generaciones, esto es un buen indicio de que

se llegó a una solución óptima. Además, podemos ver que esta estabilización de los valores del fitness ocurre justo antes de la finalización del algoritmo, por lo que, al menos para este caso, parece ser el número de generaciones indicado.

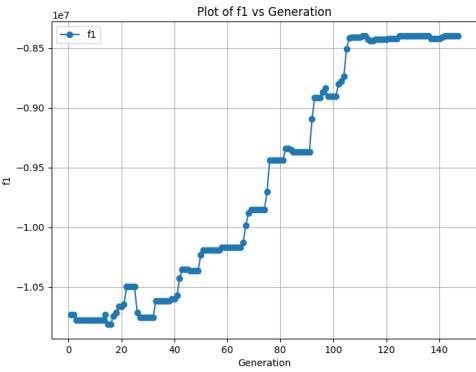


Fig. 8. Evolucion del objetivo 1

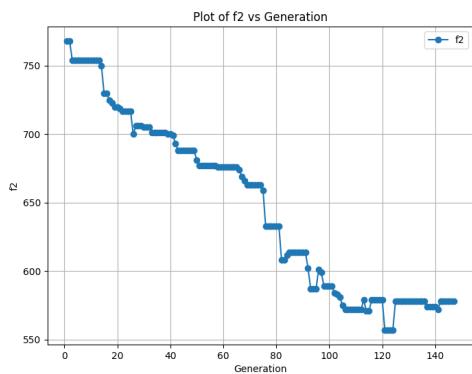


Fig. 9. Evolucion del objetivo 2

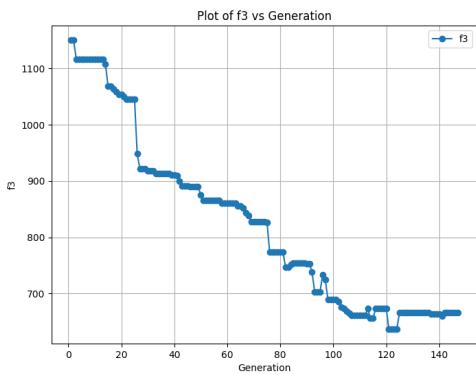


Fig. 10. Evolucion del objetivo 3

En vista de esto, parecería que el resultado obtenido cumple con los objetivos pedidos. Si proyectamos el resultado final

del algoritmo en los segmentos, veremos que el resultado parece ser coherente con una demanda real de la ciudad de Montevideo. (Ver Anexos fig. 17 y fig. 18)

Como se puede ver allí, el algoritmo le dio más importancia a la hora de ponderar los segmentos a las zonas más céntricas y transitadas, mientras que a las zonas que no poseen tanto flujo de transporte público se les dio menos relevancia. Esta demanda la podemos visualizar en la fig. 17, lo que viene a ser una representación gráfica de la función de demanda definida en la formulación matemática para la instancia de Montevideo.

Por último, se presentan los siguientes resultados para el frente de pareto aproximado:

- Mejor valor para F en el objetivo 1:

$$(-12079827, 618, 675.216)$$

- Mejor valor para F en el objetivo 2:

$$(-2862985, 279, 278.986)$$

- Mejor valor para F en el objetivo 3:

$$(-3146769, 283, 276.989)$$

- Solución ideal:

$$(-12079827, 279, 276.989)$$

- Mejor solución de compromiso:

$$(-12079827, 618, 675.216)$$

Debido a que nuestro principal objetivo es el de maximizar la demanda, es decir, que la mayor cantidad de gente se beneficie del posicionamiento de paradas, de las soluciones que provee el algoritmo, que son una variedad debido al funcionamiento del multi-objetivo, optamos por tomar como solución final la que posee el mayor f1 entre las resultantes.

A su vez, observamos que esta coincide con la mejor solución de compromiso, y es que, considerando la diferencia de magnitud entre los valores de f1 comparado con f2 y f3, tiene sentido que así sea, dado que esta solución es la que tiene una distancia euclíadiana menor con respecto a la solución ideal.

2) *Buenos Aires*: en el caso de Buenos Aires, el área metropolitana en particular, nos encontramos con que al tener un tamaño mucho mayor al de Montevideo, y con un gran número de segmentos censales, aumentó significativamente la complejidad y tiempo de ejecución del algoritmo.

Nuevamente, realizamos un frente de pareto aproximado a partir de las 20 ejecuciones del algoritmo, el cual se presenta a continuación:

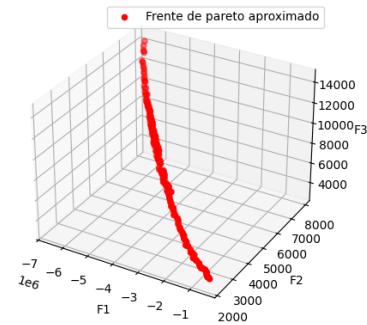


Fig. 11. Frente de pareto aproximado (BsAs)

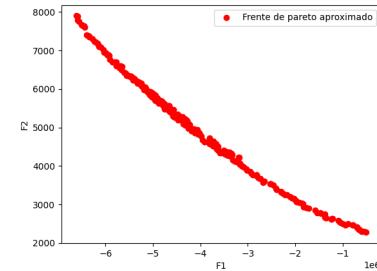


Fig. 12. Proyección en f1 y f2 del frente aproximado (BsAs)

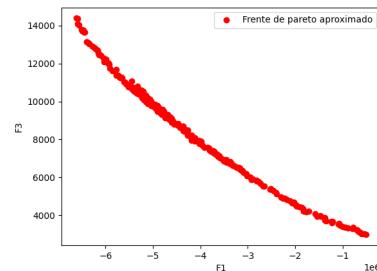


Fig. 13. Proyección en f1 y f3 del frente aproximado (BsAs)

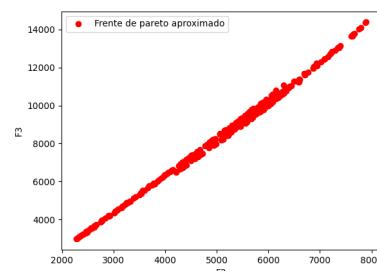


Fig. 14. Proyección en f2 y f3 del frente aproximado (BsAs)

Comparando los resultados de cada ejecución individual con el frente aproximado para la instancia de Buenos Aires (Ver Anexos: Tabla III) se llega a las siguientes conclusiones sobre las métricas:

METRICA	PROM	DESV_EST	MEJOR	P-VALOR
Tasa_error	0.9162	0.1755	0.2483	0.0315
Distancia_generacional	0.0047	0.0016	0.0008	0.7366
Spread	0.6100	0.0495	0.5119	0.9764
Hipervolumen	0.3677	0.0417	0.474	0.1964

Al igual que para la instancia anterior, nos encontramos con que la única métrica para la cual se puede rechazar la hipótesis nula es para la tasa de error. Es decir, asumimos que no sigue una distribución normal. Para el resto de las métricas no es posible rechazarla.

Finalmente, se obtuvieron los siguientes resultados para el frente de pareto aproximado:

- Mejor valor para F en el objetivo 1:

$$(-6623242, 7898, 14391.2850)$$

- Mejor valor para F en el objetivo 2:

$$(-501505, 2274, 2967.8037)$$

- Mejor valor para F en el objetivo 3:

$$(-501505, 2274, 2967.8037)$$

- Solución ideal:

$$(-6623242, 2274, 2967.8037)$$

- Mejor solución de compromiso:

$$(-6623242, 7898, 14391.2850)$$

Cabe destacar que la matriz origen destino considerada para esta instancia fue elaborada en base a los datos de un solo día, a diferencia de la de Montevideo que consideró un intervalo de tiempo mayor. Es por esto que los valores de f1 son menores en comparación con los de la instancia anterior.

Al igual que antes, se pueden ver como el algoritmo tiende a colocar paradas en las zonas con mayor demanda, cubriendo

así más efectivamente la misma. (Ver Anexos fig. 19 y fig. 20)

### C. Comparación con otras técnicas

Como mencionado al comienzo, se decidió implementar un algoritmo que sigue una estrategia greedy para fines comparativos. Este algoritmo recibe la misma matriz OD, y utiliza funciones de fitness similares. Al comienzo, la solución se toma como no poner paradas en ningún segmento censal (todos en 0), e itera sobre todos los segmentos en una lista, probando las paradas 1, 2, 3, y evaluando el fitness para cada una de esas modificaciones. Una vez termina de recorrer todos los segmentos, se queda con aquella combinación (segmento, parada) que acerque más la función de fitness a un vector ideal previamente calculado. Se agrega esta parada a la solución en el segmento correspondiente, y se procede a remover dicho segmento de la lista de segmentos restantes. Se sigue iterando de esta forma, hasta que ya no queden segmentos por considerar, o hasta que en una corrida no se encuentre ningún segmento que mejore el fitness.

Se realizó una única corrida para la instancia de Montevideo, la cual tomó 8hs 27min para terminar, y llegó al siguiente fitness final:

$$(-12222988, 987, 732.9141)$$

Como observamos anteriormente, para la primera instancia el valor del frente de pareto que más se aproximaba al vector ideal fue:

$$(-12079827, 618, 675.216)$$

El cual, para unos tiempos bastante menores de viaje y menor costo, cubre una demanda levemente menor. Además, considerando que la instancia de Montevideo es relativamente pequeña, y que el greedy se vuelve inviable para instancias de mayor tamaño, estamos satisfechos con los resultados hallados.

### D. Tiempos de ejecución

Para la instancia de Montevideo el algoritmo tomó en promedio 1581 segundos, unos 26 minutos, con una desviación estándar de 150 segundos. Por otro lado, Buenos Aires tardó 12864 segundos promedio, unas 3 horas y media. Siendo que la instancia de Buenos Aires es 24 veces mayor que la de Montevideo, la diferencia de tiempo fue mucho menos de la esperada.

## VII. CONCLUSIONES

En este informe de trabajo se buscó encontrar una combinación óptima para maximizar el uso del transporte público pero minimizando la cantidad de paradas con el fin de que se tenga que detener menos y por ende trasladarse más rápido. Además se buscó también determinar a qué segmentos habría que darle más importancia, en una escala del 1 al 3, con respecto a la calidad de las paradas y la densidad de las

mismas al ser lugares más demandados.

Lo que se puede concluir de los experimentos tomados es que los algoritmos genéticos pueden dar soluciones factibles y bastante acertadas para poder determinar en qué zonas es más conveniente colocar paradas de transporte público y en cuáles no, además se muestra bastante escalable a ciudades más grandes, por lo cuál el algoritmo puede ser usado en una variedad de diferentes ciudades independiente del tamaño de la misma.

También, consideramos que es posible mejorar la función de fitness para ajustar mejor a la demanda y costos, así como tener en cuenta otros factores como podrían ser el área del segmento considerado. Tomando todo en cuenta, los resultados obtenidos para cada instancia obtuvieron métricas razonables en cuánto a calidad de solución y diversidad al compararlo con los frentes de pareto aproximados, aun así, debido a la complejidad del problema abordado sabemos que el algoritmo planteado tiene un gran potencial de mejora.

#### REFERENCIAS

- [1] Renzo Massobrio. (2018). Urban mobility data analysis in Montevideo, Uruguay
- [2] Felipe Gonzalez, Sebastián Anapolsky. (2022). Construcción de una Matriz OD del Transporte Público para el Área Metropolitana de Buenos Aires en base a datos SUBE
- [3] Xu Xiaotong, Zheng Zhenjie, Hu Zijian, Feng Kairui, Ma Wei. (2024). A unified and validated traffic dataset for 20 U.S. cities.
- [4] Teórico del curso de Algoritmos Evolutivos de la Fing 2024

## VIII. ANEXOS

TABLE I  
RESULTADOS DE LA CONFIGURACIÓN PARAMÉTRICA

Nº	TAM_POB	CRU	MUT	HV_PROMEDIO	HV_DESVIACION_ESTANDAR	TIEMPO_PROM (seg)	P-VALOR
1	50	0.6	0.001	0.2321	0.0395	18.01	0.2243
2	50	0.6	0.01	0.3218	0.0596	17.333	0.0207
3	50	0.6	0.1	0.1538	0.0189	16.476	0.9927
4	50	0.7	0.001	0.2292	0.0581	17.925	0.0104
5	50	0.7	0.01	0.3823	0.0123	17.475	0.8641
6	50	0.7	0.1	0.1635	0.0204	16.531	0.4519
7	50	0.8	0.001	0.2414	0.0159	17.984	0.9480
8	50	0.8	0.01	0.4019	0.0129	17.714	0.3845
9	50	0.8	0.1	0.1921	0.0190	16.708	0.7912
10	100	0.6	0.001	0.2452	0.0155	18.078	0.9828
11	100	0.6	0.01	0.3985	0.0104	17.745	0.8642
12	100	0.6	0.1	0.1886	0.0148	16.771	0.9493
13	100	0.7	0.001	0.2437	0.0170	18.12	0.9807
14	100	0.7	0.01	0.3999	0.0105	17.896	0.8857
15	100	0.7	0.1	0.2167	0.0205	16.992	0.8129
16	100	0.8	0.001	0.2373	0.0206	18.155	0.9309
17	100	0.8	0.01	0.3871	0.0117	17.993	0.8541
18	100	0.8	0.1	0.2945	0.0188	17.215	0.6845
19	200	0.6	0.001	0.2265	0.0148	18.4	0.8096
20	200	0.6	0.01	0.378	0.0127	18.157	0.7992
21	200	0.6	0.1	0.2731	0.0219	17.386	0.7766
22	200	0.7	0.001	0.2294	0.0175	18.522	0.8796
23	200	0.7	0.01	0.3555	0.0172	18.33	0.6144
24	200	0.7	0.1	0.3431	0.0177	17.586	0.3001
25	200	0.8	0.001	0.2318	0.0130	18.494	0.8746
26	200	0.8	0.01	0.3231	0.0171	18.415	0.6580
27	200	0.8	0.1	0.3972	0.0071	17.91	0.4201

TABLE II  
MÉTRICAS PARA EVALUACIÓN DE RESULTADOS EN MONTEVIDEO CON RESPECTO AL FRENTE DE PARETO APROXIMADO

EJECUCION	TASA_ERROR	DISTANCIA_GENERACIONAL	SPREAD	HIPERVOLUMEN
0	0.4297	0.0012	0.5339	0.3966
1	0.9663	0.0018	0.4964	0.4458
2	1	0.0057	0.6588	0.349
3	1	0.0032	0.5911	0.3812
4	1	0.0041	0.5248	0.4627
5	1	0.0054	0.4202	0.4566
6	0.9925	0.0031	0.5914	0.4219
7	1	0.0045	0.5533	0.4142
8	1	0.005	0.4892	0.359
9	0.7051	0.0029	0.5774	0.5691
10	1	0.0033	0.6004	0.4061
11	0.7698	0.0031	0.5108	0.3948
12	1	0.0043	0.4795	0.3779
13	1	0.005	0.5525	0.4045
14	1	0.005	0.5728	0.3569
15	1	0.0033	0.5721	0.3853
16	1	0.0045	0.4993	0.4464
17	0.8654	0.0032	0.4001	0.5569
18	0.6884	0.0023	0.559	0.4133
19	1	0.004	0.4702	0.4709
20	1	0.0046	0.5437	0.3594
21	1	0.0045	0.4695	0.4964
22	1	0.0035	0.5888	0.4633
23	1	0.0055	0.5566	0.4073
24	1	0.0056	0.5586	0.3507
25	0.9578	0.0024	0.5844	0.4765
26	1	0.006	0.5996	0.3779
27	0.9651	0.0033	0.5088	0.5032
28	1	0.0048	0.4405	0.4961
29	0.9929	0.0031	0.537	0.3923

TABLE III  
MÉTRICAS PARA EVALUACIÓN DE RESULTADOS EN BUENOS AIRES CON RESPECTO AL FRENTE DE PARETO APROXIMADO

EJECUCION	TASA_ERROR	DISTANCIA_GENERACIONAL	SPREAD	HIPERVOLUMEN
0	1	0.0052	0.6161	0.3629
1	0.9482	0.0034	0.6007	0.4043
2	0.8301	0.0036	0.6091	0.4082
3	1	0.0039	0.6008	0.3753
4	1	0.0066	0.6423	0.3257
5	1	0.006	0.6678	0.3443
6	0.9929	0.0063	0.6384	0.344
7	1	0.0048	0.5906	0.3312
8	0.2483	0.0008	0.5623	0.474
9	0.7929	0.0051	0.5594	0.3332
10	1	0.0051	0.6523	0.3324
11	0.9721	0.0017	0.5832	0.4482
12	0.8307	0.0045	0.7098	0.338
13	0.938	0.0051	0.531	0.3915
14	1	0.0036	0.6905	0.3989
15	1	0.006	0.5932	0.3484
16	0.7698	0.0044	0.5119	0.3786
17	1	0.0055	0.6134	0.3349
18	1	0.0068	0.5947	0.3362
19	1	0.0062	0.6332	0.3436

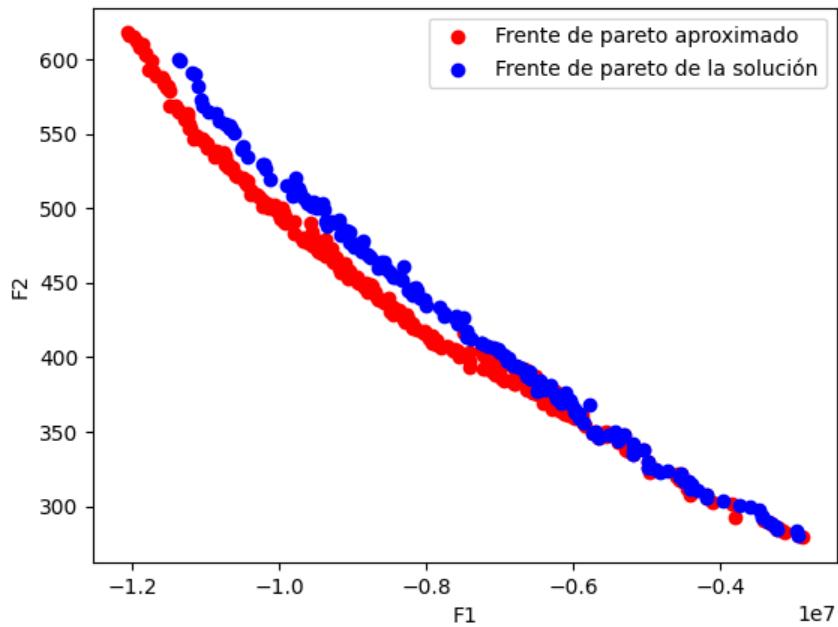


Fig. 15. Proyección en  $f_1$  y  $f_2$  de la ejecución con el mejor spread (Mvd)

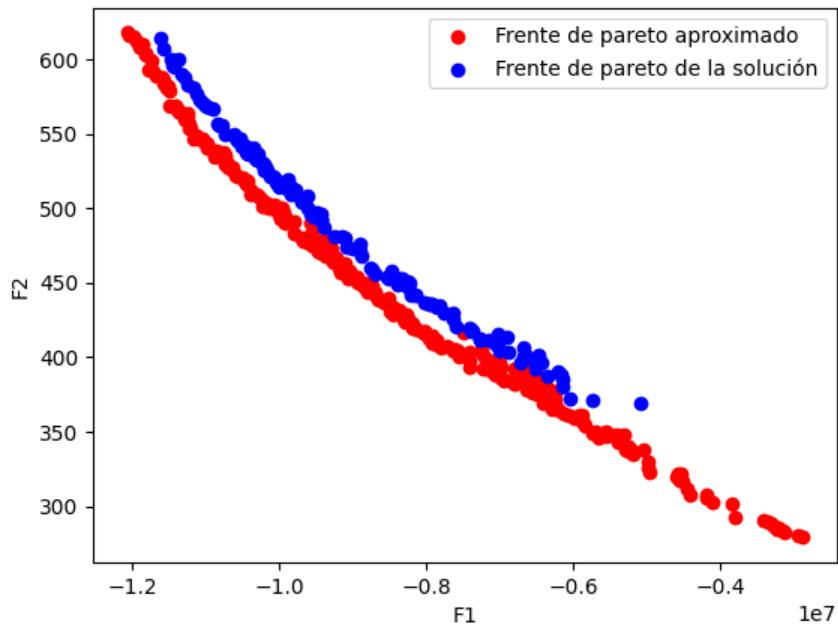


Fig. 16. Proyección en  $f_1$  y  $f_2$  de la ejecución con el peor spread (Mvd)

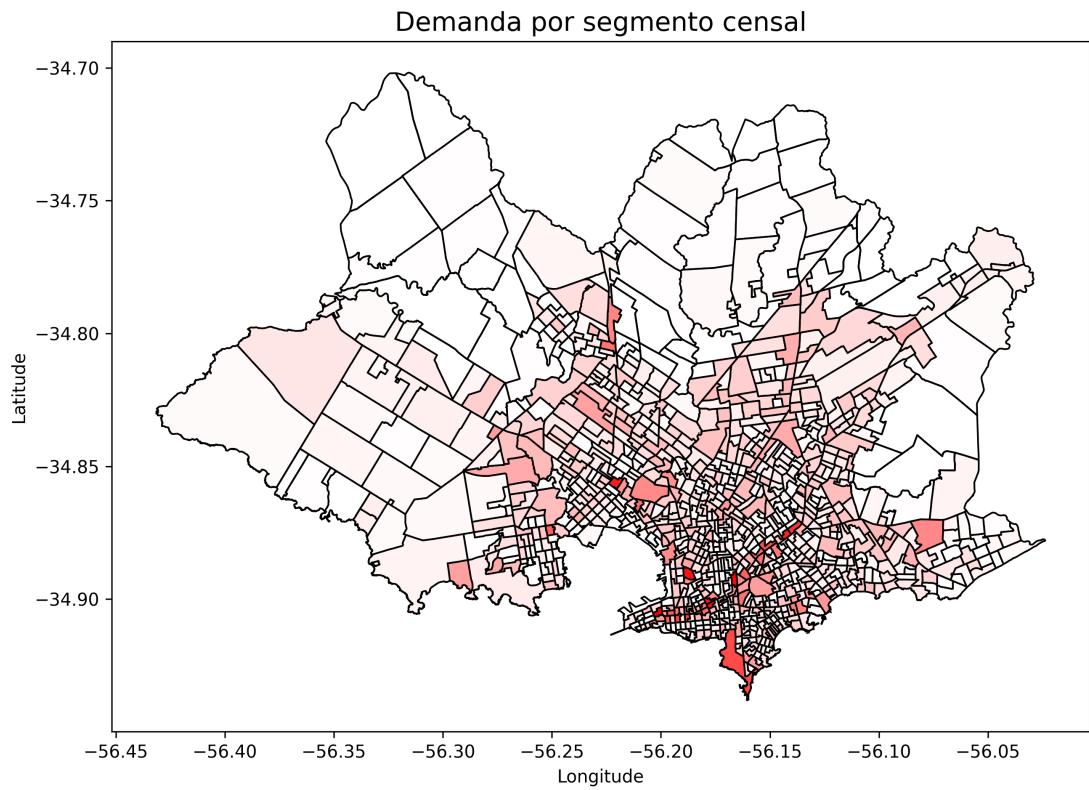


Fig. 17. Demanda por segmento en Montevideo (blanco = demanda nula, rojo = mayor demanda)

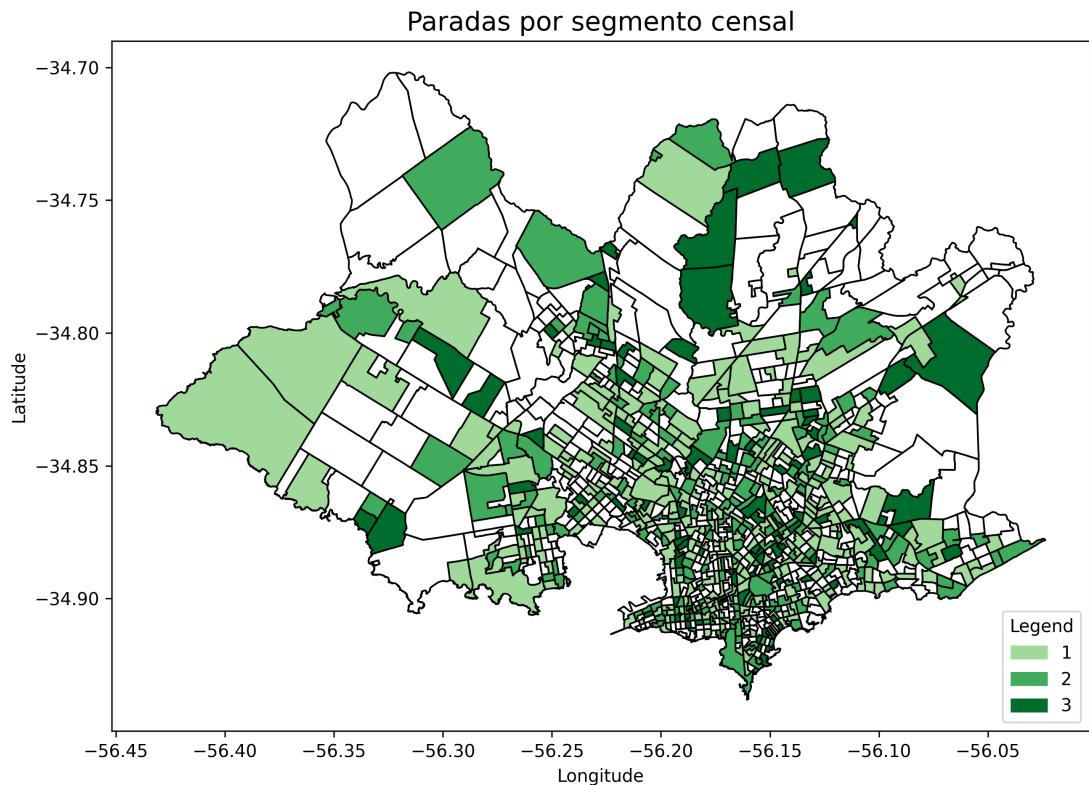


Fig. 18. Visualización de una solución en Montevideo

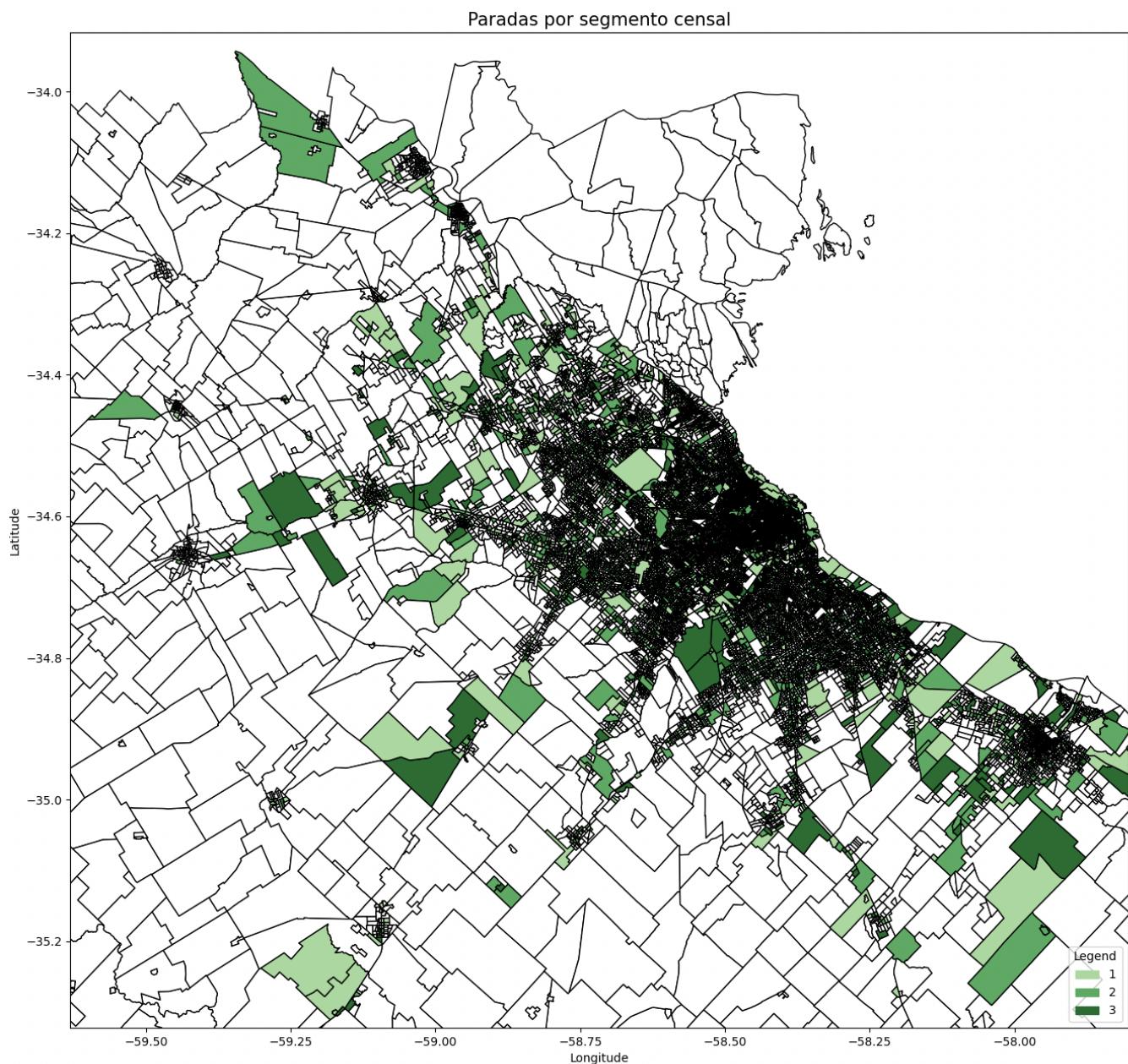


Fig. 19. Visualización de una solución en Buenos Aires



Fig. 20. Zoom en Buenos aires