ISBN: 978-628-95207-8-1. ISSN: 2414-6390. Digital Object Identifier: 10.18687/LACCEI2024.1.1.347

A Heuristic Approach to the Household Waste Collection Problem

Rodolfo Valentín-Macaya, B.S.[1], Elizabeth Montero, PhD[2] and Carola Blazquez, PhD[1]
[1]Universidad Andres Bello, Viña del Mar, Chile, r.valentnmacaya@uandresbello.edu, cblazquez@unab.cl
[1]Universidad Técnica Federico Santa María, Valparaíso, Chile, elizabeth.montero@usm.cl

Abstract- In Chile, the household solid waste collection process is a constant concern among authorities in urban areas due to its high cost and inefficient collection routes that negatively impact the environment and the population. Therefore, the objective of this study is to solve the door-to-door household solid waste collection problem using a local search algorithm, in order to obtain efficient collection routes that minimize the distance traveled by the vehicles and ensure that all streets are served, and, as a result, satisfaction of the population is increased and the proliferation of rodents and insects are prevented. In this study, three test cases with small, medium and large sizes were used in an area of the Renca commune in Santiago, Chile with four parameter configurations and three different values for the iterations and steps. The results may be used by the authorities as a decision-making tool to improve the current household solid waste collection system.

Keywords- traveling salesman problem, windy arc routing problem, waste colleting problem, simulated annealing

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

Un Acercamiento Heurístico para el Problema de Recolección de Residuos Domiciliarios Puerta-a-Puerta

Rodolfo Valentín-Macaya, B.S. [1] , Elizabeth Montero, PhD [2] , and Carola Blazquez, PhD [1]
[1] Universidad Andres Bello, Viña del Mar, Chile, r.valentnmacaya@uandresbello.edu, cblazquez@unab.cl
[2] Universidad Técnica Federico Santa María, Valparaíso, Chile, elizabeth.montero@usm.cl

***Resumen*—En Chile, el proceso de recolección de residuos sólidos domiciliarios es una preocupación constante entre las autoridades en zonas urbanas debido a su alto costo y la ineficiencia en las rutas de recolección que impactan negativamente al medio ambiente y a la población. Por lo tanto, el objetivo de este estudio es resolver el problema de recolección de residuos sólidos domiciliarios puerta-a-puerta usando un algoritmo de búsqueda local para así obtener rutas de recolección eficientes que minimicen la distancia recorrida por los vehículos y que todas las calles sean servidas, y como resultado, se aumenta la satisfacción de la población y se previene la profileración de roedores e insectos. En este estudio, se utilizaron tres casos de prueba con tamaños pequeño, mediano y grande en una zona de la comuna de Renca en Santiago de Chile con cuatro configuraciones de parámetros y tres valores de iteraciones y pasos diferentes. Los resultados se pueden utilizar como herramienta para la toma de decisiones de las autoridades para mejorar el actual sistema de recolección de residuos sólidos domiciliarios.**

***Index Terms*—problema del vendedor viajero, problema del arco ventoso, problema de recolección de residuos, recocido simulado**

I. Introduction

Chile es el país de Latinoamérica con la generación más alta de residuos sólidos municipales (456,3 kg/persona anuales). Con una población estimada de 21 millones de habitantes, Chile generó aproximadamente 8 millones de toneladas de residuos municipales en el año 2020. En la capital de Chile, Santiago, se generan aproximadamente 1.3 kg de residuos sólidos domiciliarios (RSD) por persona diarios [1], [2].

Existen diversos inconvenientes en el actual sistema de recolección de RSD puerta-a-puerta usado comúnmente en diferentes municipalidades chilenas. En algunas ocasiones, existen calles o viviendas que no son servidas por los vehículos recolectores quedando los RSD expuestos, atrayendo perros callejeros o roedores y en casos más graves exponiendo a la población a diversos tipos de enfermedades. Además, los conductores de los vehículos recolectores son responsables de establecer sus rutas de recolección, normalmente, en forma intuitiva o usando experiencia previa [3], [4]. Como resultado de estas malas prácticas, la recolección de RSD es ineficiente y genera altos costos de transporte y operacionales, reduce la satisfacción de los habitantes y genera efectos negativos en el medio ambiente y la población.

Por lo tanto, es muy relevante contar con un sistema que le permita a las municipalidades recolectar los RSD de cada vivienda y transportar de manera eficiente a rellenos sanitarios o centros de transferencia. Esto permite mantener la higiene en las viviendas de los habitantes, las calles limpias, satisfacer a la población y bajar los costos de recolección de RSD [5]. El objetivo del presente estudio es estudiar un modelo e implementar un algoritmo de búsqueda local para el problema de recolección de RSD puerta-a-puerta que permita proveer rutas a los vehículos recolectores que minimicen la distancia recorrida, mientras se priorice el tiempo y que todas las calles sean servidas.

II. Definición del Problema

El modelo matemático se basa en el modelo propuesto en [6] y que fue mejorado a partir de los modelos para el problema del cartero chino o cartero rural presentados por [7]–[10]. En la Tabla I, se lista el conjunto de parámetros principales del modelo matemático desarrollado. Sea N el total de esquinas del problema, I el conjunto de posibles nodos iniciales y F el conjunto de posibles nodos finales. Estos nodos corresponden a los puntos de entrada/salida de los vehículos a los diferentes sectores de recolección. Así, se define el conjunto $NotI$ de aquellos nodos que no pueden ser considerados nodos iniciales y $NotF$ como el conjunto de nodos que no pueden ser considerados como nodos finales. Además, se define $D_{i,j}$ como la distancia/tiempo de recorrido del arco entre el nodo i y el nodo j, el conjunto de calles bidireccionales como Abi, j y el conjunto de calles unidireccionales como Aui, j. Finalmente, sea $A = Au \cup Ab$ el conjunto de todos los arcos del problema.

Además, el modelo considera dos tipos de variables de decisión. Una variable relacionada con la cantidad de visitas de cada arco y otra variable relacionada con los nodos seleccionados como posibles puntos de inicio y fin de cada recorrido vehicular, como se muestran en (1), (2) y (3).

La función objetivo indicada en (4) busca minimizar la cantidad de veces que un vehículo recorre un conjunto de calles. Además, se puede considerar que se posee información asociada a la distancia de cada arco de calle y/o su tiempo de recorrido.

22^{nd} **LACCEI International Multi-Conference for Engineering, Education, and Technology:** *Sustainable Engineering for a Diverse, Equitable, and Inclusive Future at the Service of Education, Research, and Industry for a Society 5.0.* Hybrid Event, San Jose – COSTA RICA, July 17 - 19, 2024.

Tabla I: Parámetros del Modelo Matemático

Parámetro	Descripción
N	Total de esquinas
I	Conjunto de posibles nodos iniciales
F	Conjunto de posibles nodos finales
$NotI$	Conjunto de no posibles nodos iniciales
$NotF$	Conjunto de no posibles nodos finales
$D_{i,j}$	Distancia/tiempo recorrido del arco (i, j).
Ab	Conjunto de calles bidireccionales
Au	Conjunto de calles unidireccionales
A	Conjunto de todos los arcos del problema

Esta información puede ser incorporada en el parámetro $D_{i,j}$ asociado a cada arco. La restricciones (5) aseguran que cada arco de tipo unidireccional sea visitado al menos una vez en el sentido correspondiente. Por esto, las calles unidireccionales deben tener como mínimo un valor de 1 en sus visitas. La restricción (6) establece que todo arco bidireccional puede ser recorrido en un sentido, es decir, se puede visitar dicho arco una sola vez sin importar la dirección. La ecuación (7) controla la secuencia de visitas, sumando las entradas y salidas de cada nodo en particular. Los nodos iniciales consideran la entrada en la variable Y_i correspondiente, mientras que los nodos finales corresponde a la variable Z_i. Las restricciones (8) y (9) establecen que se debe elegir exactamente un nodo inicial dentro de las opciones correspondientes y se prohibe seleccionar un nodo no inicial. Las restricciones (10) y (11) repiten lo mismo con los nodos finales.

$$X_{i,j} = \text{Veces que se visita el arco}(i,j), \quad \forall (i,j) \in A \tag{1}$$

$$Y_i = \begin{cases} 1, \text{ si se elige } i \text{ como partida, } i \in I. \\ 0, \text{ en otro caso.} \end{cases} \tag{2}$$

$$Z_i = \begin{cases} 1, \text{ si se elige } i \text{ como término, } i \in F. \\ 0, \text{ en otro caso.} \end{cases} \tag{3}$$

$$\text{Minimizar} \sum_{(i,j) \in Arcs} D_{i,j} \cdot X_{i,j} \tag{4}$$

$$X_{i,j} \geq 1, \quad \forall (i,j) \in Au \tag{5}$$

$$X_{i,j} + X_{j,i} \geqq 1, \quad \forall i,j \in Ab \tag{6}$$

$$Y_i + \sum_{j}^{A} X_{i,j} = \sum_{j}^{A} X_{i,j} + Z_i; \forall i \in N \tag{7}$$

$$\sum_{i}^{I} Y_i = 1 \tag{8}$$

$$\sum_{i}^{NotI} Y_i = 0 \tag{9}$$

$$\sum_{i}^{F} Z_i = 1 \tag{10}$$

$$\sum_{i}^{NotF} Z_i = 0 \tag{11}$$

III. Revisión Bibliográfica

En [11], se presenta un acercamiento enfocado en la antelación a factores externos de incertidumbre que pueden afectar la recolección de RSD a nivel municipal y a nivel de una ciudad completa. Este estudio trata cada incertidumbre para poder optimizar las visitas de cada una de las calles que deben ser visitadas. Además, se presentan algoritmos y heurísticas, pero no consideran temas que sean de gran relevancia para este trabajo. El estudio de [6] es bastante similar a lo que se busca solucionar en este trabajo, ya que presenta el problema de calles sin asear y como estas afectan el diario vivir de los vecinos. Por lo tanto, se plantea y aplica el uso de un algoritmo que soluciona este problema, entregando rutas completas que recorren cada una de las calles en tiempos no mayores a los de la jornada laboral. Por lo tanto, este artículo es utilizado como referencia fundamental para el desarrollo de este trabajo. En [3], se presentan los problemas que ocurren en las calles cuando queda RSD sin recolectar y por eso se busca una nueva solución que consiste en la utilización de contenedores que quedan cerca de un sector especifico de residentes. De esta manera, se mantienen las calles limpias y se dejan los contenedores como puntos de recolección, lo que facilita la labor de los recolectores y reduce el tiempo, las rutas y las emisiones de CO2. El acercamiento de [12] está enfocado en el uso de sensores dentro de los contenedores de RSD para detectar el nivel de llenado de éstos. Así, se tiene mejor conocimiento de aquellos contenedores que necesitan ser aseados. Además, se proponen distintas estrategias para buscar aquella que permita encontrar las mejores rutas y que eviten la redundancia de visitas de contenedores que no necesitan ser aseados como también disminuir el tiempo y la distancia de las rutas de recolección. Este estudio presenta ideas bastante interesantes que pueden llegar a usarse para los casos en que el vehículo no pueda pasar por ciertas calles y se deba recurrir a los contenedores. En [13], se estudia la mejora de la recolección de RSD eco-eficientes, es decir, se busca mejorar la cantidad y la detección de RSD que pueden ser reutilizados desde contenedores de RSD. El objetivo de este artículo es utilizar algoritmos y herramientas que permitan mejorar la detección de los RSD.

IV. Propuesta de Solución

A. Transformación del problema

Esta transformación busca generar un grafo en que los arcos, convertidos en nodos, puedan ser recorridos como si el problema se tratase de un problema tipo vendedor viajero (TSP, por sus siglas en inglés). En este caso, se busca que se visiten cada uno de los arcos al menos una vez. El procedimiento toma la información correspondiente a la versión

Figura 1: Caso pequeño.

original del problema, es decir, el nodo inicial, nodo final y el costo de cada arco. Por cada línea leída, se extrae el costo de visita del nodo. Los arcos bidireccionales se transforman en dos nodos (uno en cada dirección), mientras que los unidireccionales corresponden a un único nodo. Dependiendo de si existe conexión entre arcos, desde la información de origen, se asocia el costo al arco. Si no existe una conexión entre nodos, se asocia un valor de gran magnitud con la finalidad de convertirlo en un arco demasiado costoso para que sea descartado por el proceso de minimización de costos.

Las Fig. 1 y 2 ejemplifican el proceso de transformación usando el caso de pruebas de tamaño pequeño. La Fig. 1 muestra el caso de tamaño pequeño con todos sus nodos identificados con círculos verdes y blancos, mientras que los arcos están identificados con rectángulos blancos y rojos para aquellos arcos de una sola dirección. La Fig.2 muestra el resultado de la transformación de nodos a arcos y viceversa.

B. Algoritmo Simulated Annealing

Simulated Annealing (SA) es un algoritmo de búsqueda local que se basa en el proceso de recocido de metales y cerámicas para obtener soluciones de calidad para problemas de optimización. En la homologación algorítmica, el proceso de recalentamiento de metales (SA, por sus siglas en inglés) es posible controlar la diversificación e intensificación de las soluciones encontradas, gracias a la temperatura que se utiliza como un control de calidad y de decisión para aceptar o rechazar peores soluciones en ciertos puntos del proceso. A continuación, se describen las principales componentes del algoritmo SA propuesto.

1. *Representación de soluciones:* Las soluciones se trabajan a partir de listas de visitas denominadas rutas. Una ruta consiste en una lista de nodos (o calles) ordenados según el orden de visitas.
2. *Función objetivo*: Para calcular la función objetivo, se suman todos los costos de viaje entre pares de nodos.

Algorithm 1 Simulated Annealing

1. **procedure** SIMULATED ANNEALING
2. Generar solución inicial S
3. **while** tiempoDeEjecucion **do**
4. $T = T_0 \cdot c$
5. **if** CondicionDeRecalentamiento **then**
6. $T = T_0$
7. **end if**
8. **while** IteracionesPorTemperatura **do**
9. S_{new} = movimiento(S)
10. **if** S_{new} es mejor que S **then**
11. $S = S_{new}$
12. **if** S_{new} es mejor que S_{best} **then**
13. $S_{best} = S_{new}$
14. **end if**
15. **end if**
16. **if** S_{new} no es mejor que S **then**
17. $\Delta costo = costo(S_{new}) - costo(S)$
18. **if** random(0,1) $< e^{\frac{\Delta costo}{T}}$ **then**
19. S = S_{new}
20. **end if**
21. **end if**
22. **end while**
23. **end while**
24. S_{best} = post optimización usando 2-opt
25. return S_{best}
26. **end procedure**

4. *Estructura del algoritmo:* A diferencia de un algoritmo SA tradicional, el algoritmo propuesto ha sido modificado con la finalidad de que sea capaz de permitir la repetición de visitas de nodos dentro de la lista de ruta, pues las nodos (calles) pueden ser visitados más de una vez en el problema. El pseudocódigo del SA modificado se puede observar en el algoritmo 1. En la línea 2, se genera una solución inicial, que luego iterativamente, es sometida a un proceso de aplicación de movimientos en la línea 9 y posteriormente es evaluada de acuerdo a los criterios clásicos de SA en las líneas 10 y 18. Al final del algoritmo, en la líneas 24, la mejor solución de todo el proceso s_{best} es optimizada utilizando el movimiento 2-opt.
5. *Construcción de solución inicial*: El algoritmo propuesto construye una solución inicial usando un método "Greedy" (o codicioso) que comienza eligiendo un nodo de inicio, con el cual comienza a revisar su vecindario de nodos para evaluar cuáles son los nodos vecinos con los menores costos de visita, cambia a esta posición y actualiza la ruta, continua repitiendo este proceso hasta generar una ruta completa con todos los nodos del caso.
6. *Movimientos:* El algoritmo SA propuesto implementa

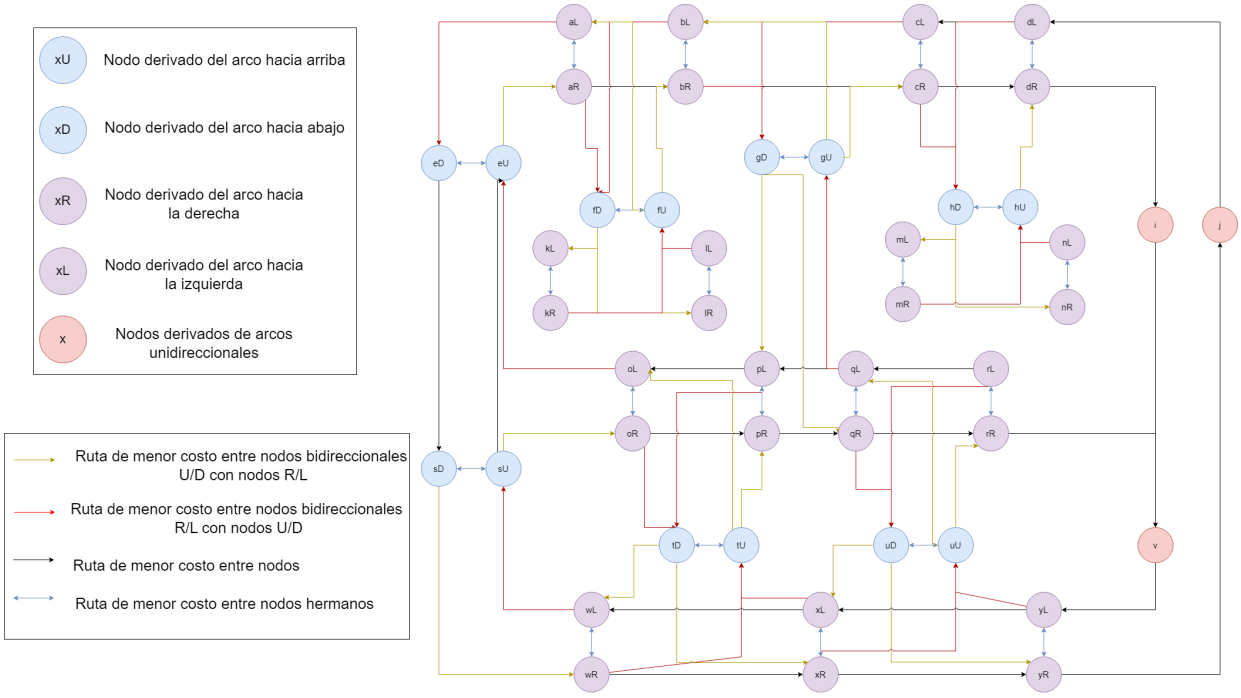

Figura 2: Version tipo TSP del caso pequeño.

dos movimientos base. Por un lado, se implementa un movimiento *swap* que intercambia los ordenes de las visitas a dos nodos (o calles). Por otro lado, el movimiento *2-opt* invierte el orden de visita de una sub-ruta perteneciente a la solución candidata. Además de estos movimientos, para este trabajo, se implementaron dos movimientos nuevos: *AgregarVisitaNodo* y *EliminarVisitaNodo*, las cuales tienen la función de agregar visitas a nodos de la ruta y eliminar visitas de nodos que posean una cantidad de visitas mayor a 1, respectivamente. Estos últimos dos movimientos se aplican probabilísticamente restringidos al valor del parámetro *Visitas*, buscando así una manera de generar un balance que permita obtener soluciones de buena calidad.

V. Configuración Experimental

A. Casos de pruebas

Los casos de pruebas fueron obtenidos mediante el software QGIS, el cual permite analizar los datos geográficamente y generar archivos de texto plano (csv) de salida con la información de arcos de regiones delimitadas.

1. *Caso pequeño:* La Fig. 3 muestra los distintos arcos que conforman el caso pequeño, y en la Fig. 4, se puede apreciar una imagen satelital de dicha porción de la comuna de Renca en la ciudad de Santiago de Chile. Este caso consiste en una pequeña sección sur de la comuna, específicamente la manzana compuesta entre las calles Chungará, Esmeralda, Topocalma y Presidente Salvador Allende. El caso considera un total de 23 intersecciones y 28 calles, de las cuales 5 son unidireccionales y 23 son bidireccionales.

Figura 3: Caso pequeño en la comuna de Renca, Santiago de Chile.

2. *Caso mediano:* Para este caso, se utilizó una zona de la parte sur de la comuna de Renca que contiene al caso pequeño y posee una mayor cantidad de sectores residenciales, lo cual es fundamental en el análisis de la recolección de RSD. La instancia cuenta de 76 nodos y 92 arcos, de los cuales 28 son unidireccionales y 64 son bidireccionales. Como es posible apreciar en

Figura 4: Imagen satelital del caso pequeño en la comuna de Renca, Santiago de Chile.

la Fig. 5, se observan los arcos considerados para ser recorridos como también se encuentran identificados los nodos con las intersecciones de calles. La Fig. 6 muestra una versión satelital del sector.

Figura 5: Caso mediano.

3. *Caso grande:* Al igual que para el caso mediano, en este caso, se considera una porción más grande de la zona sur de la comuna de Renca, con lo cual es posible generar un caso de prueba real de mayor tamaño. Este caso posee una mayor cantidad de arcos y nodos como es posible apreciar en la Fig. 7. Dentro de este caso, se encuentran los dos casos anteriores, teniendo una gran cantidad de pasajes y condominios que necesitan ser atendidos 8. Este caso de prueba considera 204 nodos y

Figura 6: Caso mediano en vista satelital.

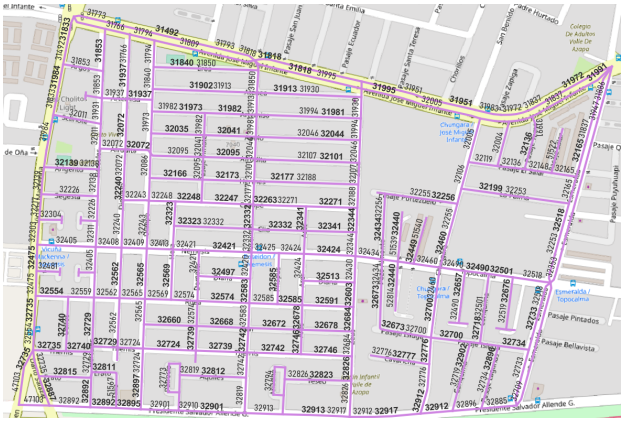

Figura 7: Caso grande.

292 arcos, de los cuales 46 son unidireccionales y 246 son bidireccionales.

B. Seteo de parámetros

En este trabajo, se realizaron pruebas con tres tiempos distintos variando los parámetros de pasos e iteraciones. La idea de estas pruebas es verificar el esfuerzo computacional requerido en la obtención de soluciones factibles y mejores funciones objetivo. Por otro lado, para cada presupuesto computacional, se analizó la relevancia del movimiento base y las probabilidades de uso de los operadores de incorporación y eliminación de nodos. Las configuraciones de los parámetros usados para cada caso y presupuesto se listan en la Tabla II.

C. Caso de pruebas pequeño

En la Tabla III, se presentan los resultados obtenidos al ejecutar el caso de prueba pequeño considerando un máximo de 30 iteraciones y 25 pasos. Los tiempos de ejecución

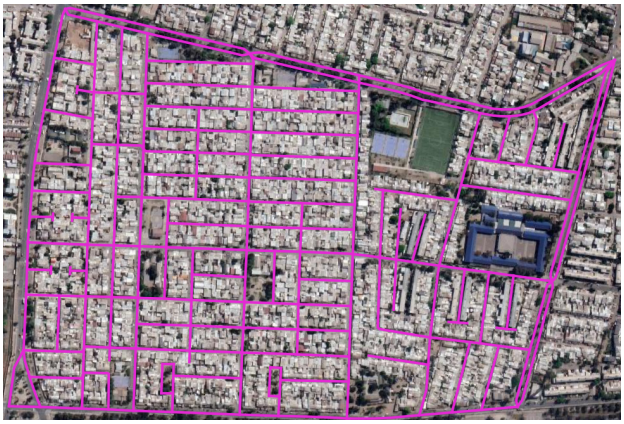

Figura 8: Caso grande en vista satelital.

Tabla II: Configuraciones de parámetros

Configuración	Movimiento base	Prob. agregar	Prob. eliminar
C1	swap	90	10
C2	2-opt	90	10
C3	swap	50	50
C4	2-opt	50	50

mostrados en esta tabla son muy bajos, menores a un minuto ya que el tamaño del caso es pequeño. En las configuraciones C1 y C2, los valores de las probabilidades de agregar y eliminar visitas son de 90 y 10 respectivamente. La configuración C1, que utiliza 2-opt como movimiento base, obtiene mejores resultados que la configuración C2. Al utilizar las configuraciones C3 y C4, se aprecia una notoria mejoría en la función objetivo de 23.08 % ya que se pasa de 21,525 como resultado con swap a 17,488 con el 2-opt. Además, la mejora porcentual entre C4-C1 es de 74.40 %, y entre C4-C2 es de 68.06 %. Esto se debe a dos razones. Primero, la configuración de agregar y eliminar visitas es de 50/50, lo que significa que se intenta agregar y eliminar una cantidad similar de visitas de nodos a la ruta en cada paso. Por otro lado, al escoger el movimiento 2-opt, se obtienen mejores resultados en búsqueda local que utilizando el movimiento swap.

En la Tabla IV, se fija un presupuesto de 100 pasos y 100 iteraciones. Al aumentar la cantidad de pasos e iteraciones, se puede notar un ligero cambio en el tiempo de ejecución entre pruebas. Al igual que en la Tabla III, se observa que las configuraciones que utilizaron el movimiento 2-opt fueron aquellas que obtuvieron los mejores resultados en términos de función objetivo. Así, las configuraciones C2 y C4 son aquellas que obtienen los mejores resultados, repitiendo la tendencia del caso pequeño 30x25.

Tabla III: Resultados del Caso Pequeño 30X25

Configuración	Función objetivo	Tiempo [min]
C1	30,499.87 ± 1,556.61	0.00 ± 0.00
C2	29,391.12 ± 1,532.47	0.00 ± 0.00
C3	21,525.00 ± 2,352.39	0.00 ± 0.00
C4	17,488.00 ± 2,184.65	0.00 ± 0.00

Si se comparan los resultados obtenidos en cada configuración es posible observar que, para la configuración 90/10 (C1-C2), se obtiene una diferencia porcentual de 39.07 %, mientras que en el caso de la configuración 50/50 (C3-C4) esta diferencia porcentual es de 147.67 %. Con esto, se concluye que en cada configuración las opciones que utilizan el movimiento 2-opt son mejores. Por su parte, al comparar las configuraciones que utilizan el movimiento swap (C1-C3), se obtiene una diferencia porcentual de 102.80 % y para el caso del movimiento 2-opt (C2-C4), se obtiene una diferencia porcentual del 261.16 %. Con estas comparaciones, es posible notar que incluso al analizar los resultados de configuraciones que utilizan los mismos movimientos, siempre 2-opt es el movimiento que presenta una mayor mejora en comparación a swap. En este sentido, al igual que en la Tabla III, es posible apreciar que los mejores resultados se obtienen en las configuraciones 50/50 utilizando como movimiento base 2-opt. Finalmente, en la Tabla V, se aumenta el presupuesto de ejecución a 500 pasos y 500 iteraciones. A partir de esta tabla ya es posible observar una diferencia en los tiempos de ejecución debida a la cantidad de pasos e iteraciones.

Tabla IV: Resultados del Caso Pequeño 100x100

Configuración	Función objetivo	Tiempo [min]
C1	18,486.37 ± 2,526.52	0.01 ± 0.00
C2	13,292.12 ± 2,935.02	0.01 ± 0.00
C3	9,115.25 ± 1,767.61	0.01 ± 0.00
C4	3,680.37 ± 927.37	0.01 ± 0.00

Tabla V: Resultados del Caso Pequeño 500x500

Configuración	Función objetivo	Tiempo [min]
C1	11,241.37 ± 1,792.62	0.30 ± 0.03
C2	6,778.12 ± 2,268.70	0.33 ± 0.03
C3	5,087.00 ± 1,490.81	0.33 ± 0.00
C4	2,440.25 ± 354.27	0.34 ± 0.00

Con respecto al desempeño de las configuraciones de parámetros estudiadas, es posible notar que, al igual que en las tablas anteriores, aquellas configuraciones que utilizaron el movimiento 2-opt obtuvieron los mejores resultados para la función objetivo. En este caso, para la configuración 90/10, la diferencia porcentual de C1-C2 es de un 65.84 %, mientras que con la configuración 50/50, la diferencia porcentual entre C3-C4 es de un 108.46 %. Luego, al comparar las configuraciones que usan los mismos movimientos, para el caso swap (C1-C3), la diferencia porcentual es de 120.98 %, mientras que en el caso 2-opt (C2-C4) esta diferencia crece a un 177.76 %. Así entonces, se replica lo presentado en la Tabla IV, donde los resultados del movimiento swap se ven superados por el movimiento 2-opt.

De estas tres tablas, es posible analizar que en cada una de las pruebas, aquellos experimentos que fueron ejecutados con el movimiento 2-opt obtuvieron como resultado funciones objetivo (en términos de costos) inferiores a los que utilizan el movimiento swap con diferencias porcentuales que van desde 23.08 % a 147.67 %. Por otro lado, el uso de probabilidades de agregación 50/50 siempre entrega mejores resultados que las probabilidades 90/10, generando mejoras porcentuales que van desde 41.69 % a 261.16 %.

D. Caso de prueba mediano

Para el caso de prueba de tamaño mediano, se realizaron experimentos con las mismas configuraciones de presupuesto computacional que para el caso pequeño. Para el mismo presupuesto, es esperable que los tiempos de ejecución resulten ser mayores debido al tamaño del caso de prueba. A continuación, se analizan las tablas con los distintos resultados obtenidos según la cantidad de pasos e iteraciones.

Tabla VI: Resultados del Caso Mediano 30x25

Configuración	Función objetivo	Tiempo [min]
C1	140,354.50 ± 2,055.89	0.01 ± 0.00
C2	141,772.25 ± 4,434.88	0.01 ± 0.00
C3	118,059.12 ± 3,689.79	0.02 ± 0.00
C4	124,677.75 ± 2,783.50	0.02 ± 0.00

Tabla VII: Resultados Caso Mediano 100x100

Configuración	Función objetivo	Tiempo [min]
C1	99,651.75 ± 3,764.01	0.22 ± 0.003
C2	99,522.75 ± 3,550.39	0.22 ± 0.005
C3	69,826.00 ± 2,294.18	0.33 ± 0.005
C4	48,690.37 ± 3,890.87	0.35 ± 0.010

Tabla VIII: Resultados del Caso Mediano 500x500

Configuración	Función objetivo	Tiempo [min]
C1	47,425.12 ± 2,894.78	27.00 ± 4.50
C2	20,663.37 ± 2,396.23	28.87 ± 4.88
C3	36,032.50 ± 2,146.58	44.50 ± 7.42
C4	10,150.75 ± 545.06	43.12 ± 7.18

La Tabla VI muestra los resultados obtenidos utilizando 30 pasos y 25 iteraciones para el caso mediano. Respecto a la función objetivo, para las configuraciones de parámetros C1 y C2 existe una ligera diferencia porcentual de solamente 1.01 %, en cambio en las configuraciones C3 y C4, las calidades obtenidas tienen una diferencia algo más marcadas con una diferencia porcentual de 5.60 %. Como resultado, el movimiento swap es el que consigue mejores resultados en términos de costos, aunque sólo por una pequeña diferencia porcentual.

Al analizar los resultados de las probabilidades de agregar es posible apreciar que cuando se usan las probabilidades 50/50, se obtienen los mejores resultados. Esto se puede corroborar al medir la diferencia porcentual entre C1 y C3 correspondiente a 18.88 % a favor de C3. Para la comparación de C2 y C4, la diferencia porcentual es de 13.71 % a favor de C4. En la Tabla VII, se pueden apreciar los resultados obtenidos para el caso mediano considerando un presupuesto computacional de 100 pasos y 100 iteraciones. Cabe destacar los tiempos de ejecución de estos resultados, los cuales indican un mayor tiempo de cómputo. En este caso, es posible observar que ninguna de las calidades medidas supera la frontera de los 100,000 en comparación con los resultados presentados en la Tabla VI.

Las configuraciones de parámetros C1 y C2 repiten su comportamiento con una diferencia de calidades bastante ajustada (0.12 % en favor de C2). Las configuraciones C3 y C4 muestran una diferencia más notoria de 43.40 % en favor de C4. En la Tabla VII, nuevamente es posible observar que los movimientos 2-opt obtuvieron un mejor desempeño con respecto a la calidad de las soluciones en comparación a lo mostrado en la Tabla VI. Esto se debe a que, si comparamos los tiempos obtenidos en la presente tabla, estos son ligeramente mayores que las configuraciones que utilizan el movimiento swap.

La diferencia porcentual de tiempo entre C1-C2 es de 0.56 %. En la misma línea, la diferencia porcentual de tiempo entre C3-C4 es de 4.16 %. Además, al comparar las probabilidad de agregar y eliminar 50/50 versus 90/10, se concluye que, para el caso de configuraciones que usan el movimiento swap (C1-C3), se obtiene una diferencia porcentual del 42.71 % a favor C3. Por otro lado, para el caso del movimiento 2-opt, la diferencia porcentual entre C2-C4 es de 104.39 % en favor de C4.

A partir de estos resultados, es posible afirmar que con el movimiento 2-opt se está utilizando un mayor tiempo de cómputo, pero se obtienen los mejores resultados en términos de calidad. En la Tabla VIII, se observan los resultados obtenidos para el caso de tamaño mediano con 500 pasos y 500 iteraciones. Comparado a los casos anteriores, se observa que los tiempos de ejecución aumentan notoriamente alcanzando tiempos de ejecución que promedian aproximadamente 35 minutos. En este caso, se obtiene una diferencia porcentual de 129.51 % al comparar C2 con C1 y de 254.97 % al comparar C4 con C3.

Respecto a las probabilidades de uso de los movimientos de agregar y eliminar se observa nuevamente que la configuración 50/50 consigue el mejor resultado. Al analizar en detalle, se obtiene que la diferencia porcentual de calidad de solución y tiempo entre C1 y C3 es de 31.61 % y de 64.81 %, respectivamente. Por otro lado, entre la configuración C2 y C4, se mide una diferencia porcentual de 103.56 % en términos de calidad y 49.35 % en tiempo de ejecución.

De todos los casos de prueba analizados, es posible afirmar que en general el movimiento 2-opt y la probabilidad de agregar 50/50 presenta las configuraciones con mejores resultados con respecto a la calidad de las soluciones. Al observar el tiempo, los mejores resultados se encuentran en el uso de una configuración 90/10.

E. Caso de prueba grande

A continuación, se presentan las tablas con los resultados obtenidos para las pruebas realizadas al caso de prueba de mayor tamaño. En este caso, se utilizaron las mismas configuraciones y parámetros que en los casos pequeño y mediano previamente revisados.

La Tabla IX muestra los resultados obtenidos en el caso grande utilizando como parámetros 30 pasos y 25 iteraciones. Al tratarse de un caso de mayor tamaño, las funciones objetivo se ven afectadas en gran medida, ya que es posible observar que todos los resultados superan los 500,000 en términos de calidad de la solución. El tiempo de ejecución también se incrementa de manera considerable consiguiendo en todas las

ejecuciones unos tiempos promedio cercanos al minuto de ejecución. Para este caso de pruebas y con esta configuraciones de esfuerzo computacional, los algoritmos que utilizaron el movimiento swap (C1 y C3) fueron los que obtuvieron un mejor desempeño en la obtención de funciones objetivo, cuya situación fue observada por primera vez en los resultados del caso mediano de la Tabla VI. Este resultado se debe a que la cantidad de pasos e iteraciones son muy bajos, el movimiento 2-opt no es capaz de encontrar una ruta de mejor calidad en un tiempo tan acotado, lo que si es logrado con el movimiento swap. Al comparar las configuraciones swap C1 y C4, se obtiene una diferencia porcentual de 0.53 % para la función objetivo y de 14.85 % para el tiempo de ejecución. Para 2-opt, en C2 y C4, se obtiene una diferencia porcentual de 1.80 % para la calidad de solución y 10.40 % para el tiempo computacional. De estos datos, es relevante observar que en cada caso los resultados swap presentaron las mejores calidades de solución, pero con diferencias bastante pequeñas. Estas pequeñas diferencias en calidad analizadas en términos de tiempo muestran diferencias porcentuales algo más marcadas.

En la Tabla X, se observan los resultados obtenidos por el algoritmo en el caso grande con una cantidad de pasos e iteraciones de 100. En esta tabla, es posible observar que los tiempos de ejecución promedian los 20 minutos. En dicho tiempo, se obtienen resultados que en su totalidad mejoran en comparación a los resultados presentados en la Tabla IX. Con respecto a la calidad de las soluciones, se nota que nuevamente las configuraciones C1 y C3 que usan el movimiento swap obtienen funciones objetivo de menor costo en comparación a C2 y C4, que emplean el movimiento 2-opt. En este caso, se presentan diferencias porcentuales de 24.18 % y 20.39 %, respectivamente. Para las ejecuciones de C1 y C3, se obtiene una diferencias porcentual en el tiempo computacional de 52.90 %, y para las configuraciones C2 y C4, se obtiene una diferencia de 30.99 %.

Tal como en casos anteriores de las Tablas VI y IX, se observan diferencias entre las ejecuciones 50/50 y 90/10. Las configuraciones que utilizan 2-opt se ven superadas por sus contrapartes que utilizan swap. Analizando las configuraciones C1 y C3, se obtiene una diferencia porcentual del 24.18 % para sus calidades y 52.90 % para sus tiempos de ejecución. Por otro lado, para C2 y C4, se obtiene una diferencia porcentual de 20.39 % para las calidades de solución y de 30.99 % para los tiempos de cómputo.

Tabla IX: Resultados del Caso Grande 30x25

Configuración	Función objetivo	Tiempo [min]
C1	524,392.25 ± 3,225.61	1.06 ± 0.02
C2	527,204.62 ± 2,679.86	1.22 ± 0.07
C3	509,013.12 ± 5,641.56	1.87 ± 0.34
C4	518,192.50 ± 2,994.13	1.69 ± 0.34

En la Tabla XI, se aprecian los resultados obtenidos con el algoritmo propuesto considerando 500 pasos y 500 iteraciones en las ejecuciones. Nótese que debido a la excesiva cantidad de tiempo que toma realizar cada una de las ejecuciones

Tabla X: Resultados del Caso Grande 100x100

Configuración	Función objetivo	Tiempo [min]
C1	477,116.12 ± 4,365.03	15.35 ± 0.76
C2	499,599.50 ± 3,913.88	18.39 ± 0.34
C3	384,189.37 ± 5,155.94	23.48 ± 1.21
C4	414,980.50 ± 8,810.08	24.08 ± 0.72

Tabla XI: Resultados Caso Grande 500x500

Configuración	Función objetivo	Tiempo [min]
C1	266352	365
C2	244295	742
C3	193049	547
C4	88545	1110

(superior a 6 horas de ejecución), se decidió agregar sólo una ejecución por configuración. Las configuraciones C2 y C4 son aquellas que utilizan mayor tiempo de ejecución, donde la configuración C4 requirió un total de 18 horas de ejecución. Esta gran cantidad de tiempo invertido en estas pruebas se ve bien reflejado en la obtención de calidades en términos de funciones objetivo. Independiente del movimiento utilizado, los resultados obtenidos están por debajo de la franja de costos de 270,000, los cuales son muy bajas en comparación a los resultados presentados en las Tablas IX y X. Al comparar los resultados entre las Tablas XI y IX, se obtiene una diferencia porcentual promedio de 106.93 % y 2.38 % para la calidad de la solución y tiempo de ejecución, respectivamente. En cambio, al comparar los resultados con la Tabla X, la diferencia es igual a 106.93 % en la calidad de solución y 22.28 % en tiempo de cómputo.

Al comparar los resultados de las funciones objetivo obtenidos por cada configuración, se nota que esta vez los casos que usan el movimiento 2-opt (C2 y C4) son aquellos que obtienen el mejor rendimiento, pero con tiempo computacional bastante alto. Cuando se comparan los resultados obtenidos entre C1 y C2, se obtiene una diferencia porcentual de 9.02 %, mientras que en el caso de C3 y C4, se obtiene una diferencia porcentual de 118.02 %. En términos de tiempo, para C1-C2, se obtiene una diferencia porcentual de 103.28 %, mientras que para C3-C4, este valor es de 102.92 %, siendo estas diferencias bastante similares.

Al revisar las configuraciones que utilizan el movimiento swap, se obtiene una diferencia porcentual entre C1 y C3 de 37.97 % y una diferencia de tiempo de 49.86 %. Para los casos que utilizan 2-opt, C2 y C4 obtienen una diferencia porcentual en términos de calidad de 175.89 %, y en términos de tiempo de ejecución de 49.59 %. De esta manera, los mejores resultados con respecto a las funciones objetivo se encuentran en las configuraciones que utilizan 50/50 para sus parámetros e iteraciones. Las diferencias porcentuales en tiempo son bastante similares, pese a ésto, se observa que una configuración 90/10 siempre presentará un valor menor en el tiempo de ejecución.

En general, se observaron comportamientos bastante interesantes para ambos tipos de movimientos con respecto a las tablas anteriores ya que se mostró que las ejecuciones realizadas con el movimiento swap obtuvieron una calidad

superior al movimiento 2-opt. En casos anteriores, se trabajó con casos de pruebas pequeños y medianos, donde la cantidad de nodos no supera los 80. Esto sumado a la cantidad de iteraciones y pasos puede resultar en tiempos mayores, pero con resultados en la función objetivo de mejor calidad. En la Tabla XI, se encontraron valores de calidad bastante bajos para el movimiento 2-opt usando tiempos de ejecución excesivamente largos (superando las 4 horas de ejecución).

En resumen, se concluye que si el caso de prueba es pequeño y la cantidad de iteraciones y pasos superan las centena, es seguro encontrar que el movimiento 2-opt es el mejor para encontrar calidades mínimas, pero con tiempos de ejecución mayores que el movimiento swap. A medida que el caso de estudio crece en tamaño, 2-opt entrega soluciones bastante discutibles, como se observa en las Tablas IX y X, donde el tiempo de ejecución es mayor con una función objetivo de mejor calidad. Para casos de gran tamaño con un alto número de pasos e iteraciones, se obtienen mejores valores para los casos que utilizan el movimiento 2-opt.

F. Análisis de soluciones

En esta sección, se presentan figuras con algunas soluciones obtenidas al aplicar el algoritmo SA a los casos propuestos. En estas figuras, los arcos verdes representan arcos no permitidos (i.e., arcos no pueden ser recorridos por su sentido), los arcos rojos son aquellas conexiones entre nodos que no son posible de recorrer y que fueron marcados en los casos de pruebas con valores muy altos, y los arcos negros corresponden a conexiones entre nodos que cumplen las restricciones y que son posibles de recorrer.

La Fig. 9 muestra la mejor solución obtenida al ejecutar el algoritmo SA con los parámetros 30x25 con el movimiento 2-opt. Todos los nodos son visitados al menos una vez en la solución. Para este caso, la calidad que se obtiene de la ruta calculada en SA es de 14,124. El nodo de inicio (marcado con una I) es el 5 y el nodo final (marcado con F) es el 15. En esta figura, existe una gran cantidad de arcos rojos y verdes debido al poco tiempo de cómputo del algoritmo. Al ser un caso tan pequeño y con una cantidad de iteraciones y pasos tan baja, no se alcanza el mejor resultado. La ruta resultante posee conectividad entre nodos que son imposibles de recorrer en un caso real (e.g., nodo 0 al nodo 48). Además, es posible notar que en muchos casos las conexiones con arcos verdes no deberían existir, tal como la conexión del nodo 36 con dirección sur y el nodo 21 con dirección norte, en el lado izquierdo de la figura.

En la Fig. 10, se muestra la mejor ruta obtenida para el caso pequeño de 100x100, donde también se utiliza la configuración 50/50 y el movimiento 2-opt. Para este caso, la calidad de la solución con el algoritmo SA es de 2,495. Al igual que en el caso 30x25, todos los nodos son visitados, pero para este caso el nodo 5 es visitado más de una vez, el cual se visita a sí mismo una vez para luego visitar al nodo 7 y seguir la ruta. Esto muestra a simple vista que la ruta realiza una visita innecesaria, pero también sirve como muestra de la elección de rutas que realiza el algoritmo y la evaluación de los nodos a conectar. El punto de inicio de la ruta obtenida en la Fig. 10 está situado en el nodo 9, y punto de término en el nodo 2. Además, es posible apreciar que, a diferencia del caso anterior, no existen arcos de color rojo, pero si hay nuevamente una gran cantidad de conexiones de color verde, es decir, arcos no permitidos y un total de 12 arcos factibles de color negro. En dicha figura, se muestra que los nodos 8, 23 y 37 tienen dirección hacia el norte, pero los arcos verdes están en el sentido inverso, lo que muestra una conexión incorrecta.

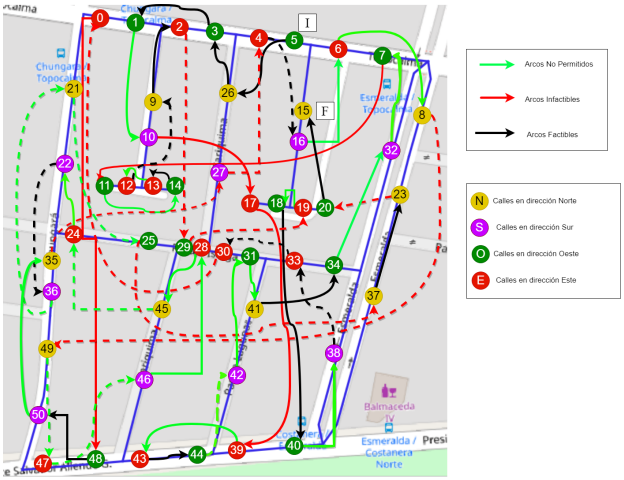

Figura 9: Ruta SA - Caso pequeño - 30x25

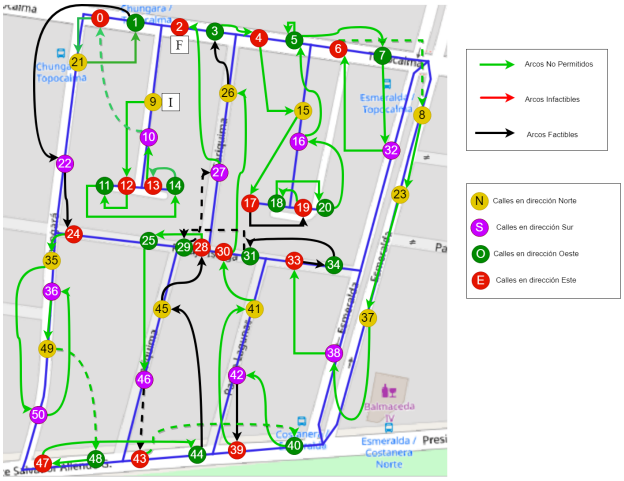

Figura 10: Ruta SA - Caso pequeño - 100x100

En la Fig. 11, se observa la solución obtenida por SA para el caso pequeño de 500x500 utilizando los mismos parámetros que los dos casos anteriores, generando como resultado un solución de 2,246. En este caso, se marcan como nodo de inicio al nodo 3, y como nodo final de la ruta al nodo 27. De manera, similar a los casos anteriores, se recorren cada uno de los nodos a visitar. A diferencia del caso pequeño de 100x100, en este caso, no hay nodos que se visitan en más de una ocasión, si no cada nodo es visitado solo una vez.

Al igual que en el caso pequeño de 100x100, en esta ruta, no existen arcos infactibles, probablemente gracias a la cantidad

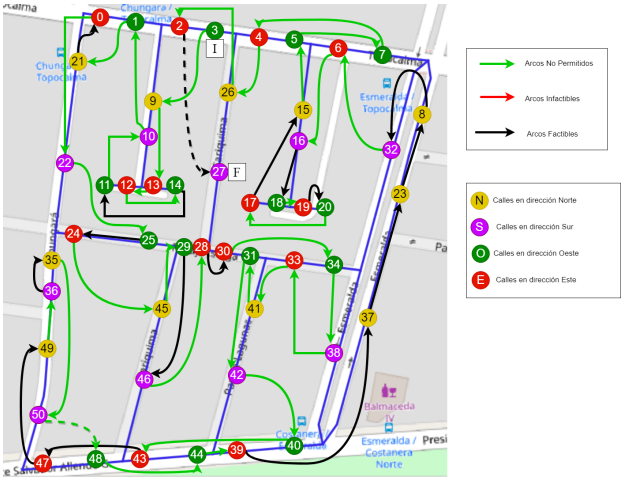

Figura 11: Ruta SA - Caso pequeño - 500x500

de pasos e iteraciones de estos casos de pruebas, lo que permite al algoritmo tener más tiempo y calcular así una mayor cantidad de veces las mejores rutas que este pueda encontrar. Además, se obtienen arcos verdes nuevamente y esta vez los arcos negros aumentan su cantidad a 14 arcos factibles.

A partir de los mapas expuestos es posible apreciar que, a medida que los casos van creciendo en presupuesto respecto a pasos e iteraciones, es posible ver que la cantidad de arcos infactibles (verdes) comienza a disminuir hasta casi desaparecer completamente. Al haber una mayor cantidad de pasos para analizar, el algoritmo puede explorar una mayor cantidad de veces las mejores rutas que tiene a su disposición, lo que implica un mayor tiempo de cómputo, pero también obtiene una mejora en las calidades de las soluciones. En esta sección, se obtuvo una solución de 14,124 con un presupuesto de 30x25, luego se bajó significativamente a un 2,495 utilizando un presupuesto de 100x100, y finalmente se obtuvo una calidad de 2,246 con un presupuesto 500x500. Las mayores reducciones en términos de calidad fueron conseguidos en su mayoría gracias a los arcos no permitidos de color verde.

VI. Conclusiones

Este estudio tiene como objetivo resolver el problema de recolección de residuos domiciliarios puerta-a-puerta usando el algoritmo Simulated Annealing. Para ello, se utilizaron casos de pruebas de tres tamaños diferentes proveniente de datos reales de la comuna de Renca en Santiago de Chile. Además, se probaron distintos parámetros pasos e iteraciones del algoritmo y se evaluó la relevancia del movimiento base (2-opt y swap) y las probabilidades de uso de los operadores de incorporación y eliminación de nodos (calles) en los grafos creados.

En general, los resultados sugieren que a medida que aumenta el número de iteraciones y pasos, el tiempo de ejecución se incrementa y la calidad de la solución mejora al disminuir la función función objetivo, en términos de costos. Con respecto, a las configuraciones de los parámetros, para casos de prueba pequeños y medianos, se concluye que el movimiento 2-opt tiende a presentar mejores resultados que el movimiento swap, mientras que el movimiento swap entrega mejores resultados para casos de prueba de tamaño grande. Además, las probabilidades 50/50 de agregar y eliminar nodos presentan mejores resultados que las probabilidades 90x10 para los tres tamaños de casos de prueba.

Como futura investigación, sería interesante explorar el uso de otros algoritmos meta-heurísticos para resolver el problema tales como algoritmos basados en colonias de hormigas que han demostrado buenos resultados en problemas de ruteo y, además permitirían lidiar con la conectividad de rutas de manera más directa sin la necesidad de transformar el problema. Además, se deberían considerar problemas de mayor tamaño y el uso de algoritmos de segmentación que determinen las mejores zonas para recolección de RSD. Por último, sería interesante resolver casos de prueba de otras comunas de Chile para analizar nuevos desafíos que puedan surgir en el problema.

Agradecimientos

Este trabajo fue financiado por ANID con el proyecto FONDECYT 1241471 y el proyecto FONDEF ID22I10107.

Referencias

[1] Cifelli, Rafaella, “Conoce las cifras de reciclaje en chile por tipo de residuo.” [Online]. Available: https://codexverde.cl/conoce-las-cifras-de-reciclaje-en-chile-por-tipo-de-residuo/

[2] Biblioteca del Congreso Nacional de Chile. (2023) Residuos: Conceptos, datos de residuos generados y regulación nacional. [Online]. Available: https://obtienearchivo.bcn.cl/

[3] C. Blazquez and G. Paredes-Belmar, “Network design of a household waste collection system: A case study of the commune of Renca in Santiago, Chile,” *Waste Management*, vol. 116, pp. 179–189, 2020.

[4] C. Letelier, C. Blazquez, and G. Paredes-Belmar, “Solving the bin location–allocation problem for household and recycle waste generated in the commune of renca in santiago, chile,” *Waste Management & Research*, vol. 40, no. 2, pp. 154–164, 2022.

[5] C. A. Arribas, C. A. Blazquez, and A. Lamas, “Urban solid waste collection system using mathematical modelling and tools of geographic information systems,” *Waste Management & Research*, vol. 28, no. 4, pp. 355–363, 2010.

[6] G. Braier, G. Durán, J. Marenco, and F. Wesner, “An integer programming approach to a real-world recyclable waste collection problem in Argentina,” *Waste Management Research: The Journal for a Sustainable Circular Economy*, vol. 35, no. 5, pp. 525–533, 2017.

[7] F. Ferreira Gomes, “Optimization of routes for road surface inspection of a portuguese national road network,” *Instituto Superior Técnico Lisboa*, 09 2015.

[8] G. Groves and J. Van Vuuren, “Efficient heuristics for the Rural Postman Problem,” *ORiON*, vol. 21, no. 1, 2005.

[9] M. Drexl, “On the Generalized Directed Rural Postman Problem,” *Johannes Gutenberg University*, 09 2012. [Online]. Available: https://logistik.bwl.uni-mainz.de/files/2018/12/LM-2012-03.pdf

[10] H. A. Eiselt, M. Gendreau, and G. Laporte, “Arc Routing Problems, Part II: The Rural Postman Problem,” *Operations Research*, vol. 43, no. 3, pp. 399–414, 1995.

[11] Y. Li and G. Huang, “Modeling Municipal Solid Waste Management System under Uncertainty,” *Journal of the Air Waste Management Association*, vol. 60, no. 4, pp. 439–453, 2010.

[12] T. R. P. Ramos, C. S. de Morais, and A. P. Barbosa-Póvoa, “The smart waste collection routing problem: Alternative operational management approaches,” *Expert Systems with Applications*, vol. 103, pp. 146–158, 2018.

[13] A. Expósito-Márquez, C. Expósito-Izquierdo, J. Brito-Santana, and J. A. Moreno-Pérez, “Greedy randomized adaptive search procedure to design waste collection routes in La Palma,” *Computers Industrial Engineering*, vol. 137, p. 106047, 2019.