

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“RUTEO DE VEHÍCULOS DE RECOLECCIÓN DE RESIDUOS PUERTA A PUERTA CON VENTANAS DE TIEMPO”

BENJAMÍN GUZMÁN LÓPEZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Elizabeth Montero
Profesor Correferente: Nicolás Rojas-Morales

Enero - 2024

DEDICATORIA

A mis queridos padres, por su inquebrantable apoyo, dedicación y amor, así como por cada sacrificio que han realizado a lo largo de estos años. Con profundo cariño y gratitud,

Benjamín Guzmán López.

AGRADECIMIENTOS

Quiero expresar mi sincero agradecimiento a mi familia, cuyo respaldo ha sido el pilar fundamental durante estos años académicos. Su apoyo incondicional y los sacrificios que han realizado han sido la base de mi crecimiento y desarrollo.

A mis compañeros de clase, les agradezco por enfrentar juntos los desafíos académicos y por todo lo que hemos aprendido mutuamente. Los recuerdos que hemos creado serán tesoros que guardaré siempre.

A mi profesora guía, le agradezco sinceramente por su orientación, dedicación y paciencia durante todo el proceso de elaboración de esta memoria. Sus consejos y comentarios han sido cruciales para dar forma a mi trabajo.

RESUMEN

Resumen— El problema del ruteo de vehículos de recolección de residuos consiste en obtener la ruta o camino más corto que permita a los recolectores de basura recolectar todos los desechos de las personas. La variante del problema que se estudiará corresponde a la versión donde los camiones recolectores de desechos deben pasar por todas las casas de un sector y están limitados por ventanas de tiempo que restringen las rutas que pueden realizar los vehículos. En este trabajo de memoria se propone un algoritmo de búsqueda heurística para encontrar los caminos en tiempos acotados para casos reales del problema.

Palabras Clave— Problema del Cartero Chino Rural, Problema del Cartero Chino, Heurísticas, Problema de enrutamiento de arcos

ABSTRACT

Abstract— The problem of waste collection vehicle routing consists of obtaining the shortest route or path that allows garbage collectors to collect all waste from the residents. The variant of the problem to be studied corresponds to the version where waste collection trucks must pass through all houses in a given area and are constrained by a time windows that restrict the routes the vehicles can take. This thesis proposes a heuristic search algorithm to find paths within bounded time limits for real-world instances of the problem.

Keywords— Rural Postman Problem; Chinese postman problem; Heuristics; Arc Routing Problem

GLOSARIO

- ARC: *Arc Routing Problem*
- VRP: *Vehicle Routing Problem*
- GTSP: *Generalized Travelling Salesman Problem*
- CARP: *Capacited Arc Routing Problem*
- TS: *Tabu Search*
- CPP: *Chinese Postman problem*
- RPP: *Rural Postman Problem*
- GRP: *General Routing Problem*
- DGRP: *Directed general routing problem*

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE TABLAS	IX
ÍNDICE DE PSEUDOCÓDIGOS	X
INTRODUCCIÓN	1
1 Objetivos	2
1.1 Objetivo General	2
1.2 Objetivos específicos	2
CAPÍTULO 2: DEFINICIÓN DEL PROBLEMA	3
2.1 Modelo Matemático	4
2.1.1 Parámetros	4
2.1.2 Notación	4
2.1.3 Variables de decisión	5
2.1.4 Función Objetivo	5
2.1.5 Restricciones	5
2.2 Resumen	7
CAPÍTULO 3: ESTADO DEL ARTE	8
3.1 Problema del cartero chino	8
3.1.1 Problemas con grafos no dirigidos	9
3.1.2 Problemas con grafos dirigidos	9
3.1.3 Problemas con grafos mixtos	10
3.1.4 Problemas con grafos ventosos	11
3.1.5 Combinación de variantes	12
3.2 Problemas de recolección de desechos	12
3.3 Problema de enrutamiento de arcos	17
3.4 Resumen	21
CAPÍTULO 4: PROPUESTA DE SOLUCIÓN	22
4.1 Representación de las soluciones	22
4.2 Función de evaluación	24
4.2.1 Tiempos de viaje	24
4.2.2 Penalización por ventana de tiempo	25

4.3 Estructura	26
4.3.1 Inicialización	27
4.3.2 Construcción de soluciones	27
4.3.3 Actualización de feromona	30
4.3.4 Criterios de detención	31
4.3.5 Resumen	31
CAPÍTULO 5: VALIDACIÓN DE LA SOLUCIÓN	33
5.1 Experimentación	33
5.2 Sintonización de parámetros	34
5.2.1 Resultados Sintonización	34
5.3 Instancias de prueba	35
5.3.1 Formato	35
5.3.2 Generación de casos	36
5.3.3 Caso 1: Renca	37
5.3.4 Caso 2: Cerro Navia	38
5.3.5 Caso 3: Quinta Normal	39
5.4 Resultados	41
5.4.1 Caso 1: Renca	43
5.4.2 Caso 2: Cerro Navia	47
5.4.3 Caso 3: Quinta Normal	49
5.5 Resumen	51
CAPÍTULO 6: CONCLUSIONES	53
REFERENCIAS BIBLIOGRÁFICAS	56

ÍNDICE DE FIGURAS

1	Generación de una nueva ruta luego de eliminar 2 arcos	14
2	Ejemplo de enrutamiento de arcos con la incorporación de nuevos nodos (los costos originales de los arcos se muestran junto a cada uno de ellos). Fuente: [Pearn <i>et al.</i> , 1987]	18
3	Ejemplo de una solución utilizando la representación propuesta	22
4	Ejemplo de una instancia pequeña para el problema del ruteo de vehículos de recolección de residuos puerta a puerta con ventanas de tiempo. Fuente: Propia	23
5	Grafo de una instancia con 2 nodos y un arco. Fuente: Propia	26
6	Ejemplo de una hormiga buscando arcos pendientes utilizando su memoria a corto plazo. Fuente: Propia	29
7	Ejemplo de una hormiga deshaciendo un movimiento utilizando su memoria a corto plazo. Fuente: Propia	30
8	Formato instancias de pruebas	35
9	Ejemplo de instancia de prueba de cuatro nodos	36
10	Sección 1 (tamaño pequeño)	37
11	Sección 2 (tamaño mediano)	38
12	Sección 3 (tamaño grande)	39
13	Sección 3 (tamaño pequeño)	40
14	Sección 3 (tamaño mediano)	41
15	Sección 3 (tamaño grande)	42
16	Quinta Normal: Sección 1 (tamaño pequeño)	43
17	Quinta Normal: Sección 2 (tamaño mediano)	44
18	Sección 3 (tamaño grande)	45

ÍNDICE DE TABLAS

1	Resumen de las variantes	12
2	Tasa de generación de basura en el periodo 1	15
3	Tasa de generación de basura en el periodo 2	15
4	Tasa de generación de basura en el periodo 3	15
5	Flujo de basura	16
6	Representación de la solución de la instancia de ejemplo Fuente: Propia	24
7	Parámetros y sus valores posibles	34
8	Tabla de comparación entre las dos configuraciones seleccionadas.	35
9	Resumen soluciones: Comuna de Renca	46
10	Resumen estadísticas: Comuna de Renca	46
11	Resumen tiempos de ejecución: Comuna de Renca	47
12	Resumen soluciones: Comuna de Cerro Navia	49
13	Resumen estadísticas: Comuna de Cerro Navia	49
14	Resumen tiempos de ejecución: Comuna de Cerro Navia	49
15	Resumen soluciones: Comuna de Quinta Normal	51
16	Resumen estadísticas: Comuna de Quinta Normal	51
17	Resumen tiempos de ejecución: Comuna de Quinta Normal	51

ÍNDICE DE ALGORITMOS

1 Algoritmo Propuesto	26
2 find path	29
3 Actualización de feromonas	31

INTRODUCCIÓN

En la actualidad, el proceso de recolección de desechos constituye aproximadamente hasta el 70 % del costo total asociado al procesamiento de desechos, subrayando la crítica relevancia de la logística detrás de este proceso. La eficiencia y eficacia de este procedimiento se han convertido en aspectos de gran importancia para reducir costos y, más recientemente, para mitigar la contaminación generada por estos vehículos.

El proceso de recolección de desechos puede abordarse como un problema de ruteo de vehículos, específicamente como el problema del cartero chino, que consiste en encontrar el camino más corto que visite cada arco de un grafo al menos una vez, regresando al nodo inicial después de recorrer todos los arcos.

En este trabajo, se propone un algoritmo basado en colonias de hormigas que incorpora memoria a corto plazo. El objetivo es que las hormigas encuentren una ruta que abarque todas las calles de una región específica. Además, el algoritmo estará sujeto a una ventana de tiempo, lo que implica que las hormigas deben hallar una ruta dentro de un límite temporal, y en caso de no cumplir con este límite, serán penalizadas.

La introducción de la memoria a corto plazo en las hormigas tiene como objetivo facilitar su desplazamiento por calles ya recolectadas, mejorando la eficiencia al evitar la repetición de la recolección. Este mecanismo simula que el vehículo ya ha completado la recolección en la calle correspondiente, de modo que la próxima vez que pase por la misma ubicación, simplemente se desplazará sin realizar nuevamente la recolección.

El algoritmo propuesto se evaluará con distintos parámetros en tres comunas de Santiago de Chile: Renca, Cerro Navia y Quinta Normal. Cada comuna será evaluada en tres zonas distintas y de distinto tamaño. Además, cada caso se evaluará con una ventana de tiempo diferente, por lo que en total, el algoritmo será evaluado con 54 casos para cada conjunto de configuración.

Las comunas elegidas destacan por su atractivo geográfico y su interesante topografía vial. Cuentan con calles de diversos tipos y tamaños, así como rotondas e intersecciones complejas, lo que añade un nivel de complejidad a la generación de soluciones. Adicionalmente, la selección de estas comunas no solo se basa en sus características viales, sino también en su proximidad geográfica. Esta elección estratégica permitirá, en futuros trabajos, analizar la viabilidad de implementar sistemas de recolección de desechos intercomunales con el fin de optimizar aun más este proceso.

La estructura del documento se divide en cinco capítulos, cada uno desempeñando un papel crucial en la comprensión y desarrollo de la investigación. El primer capítulo se centra en la definición del problema, proporcionando un contexto claro y específico que establece las bases para el trabajo.

El segundo capítulo realiza un análisis del estado del arte, abordando tanto el problema específico como problemas similares. También presenta un análisis histórico de los problemas de recolección de desechos.

El tercer capítulo expone la propuesta de solución, detallando el algoritmo basado en colonias de hormigas y exponiendo los fundamentos teóricos y la lógica subyacente.

El cuarto capítulo se centra en la validación del algoritmo, utilizando casos reales basados en tres comunas de Santiago de Chile para evaluar el desempeño del algoritmo y la eficacia y eficiencia en estas condiciones.

Finalmente, el quinto capítulo presenta las conclusiones derivadas de los resultados obtenidos, resumiendo las contribuciones, limitaciones y ofreciendo sugerencias para investigaciones futuras.

A continuación se listan los objetivos de este trabajo de memoria.

Objetivos

1.1. Objetivo General

Proponer una herramienta basada en una estrategia de búsqueda heurística para el problema del ruteo de vehículos de recolección de residuos puerta a puerta con ventanas de tiempo.

1.2. Objetivos específicos

- Reconocer y contextualizar el problema dentro del campo de la inteligencia artificial, describiendo sus variantes y soluciones más conocidas a lo largo del tiempo.
- Proponer una técnica de búsqueda heurística que sea capaz de encontrar soluciones de calidad para el problema en tiempos de cómputo acotados.
- Analizar y evaluar el algoritmo propuesto verificando su desempeño en términos de calidad de soluciones y tiempo con instancias reales.

CAPÍTULO 2

DEFINICIÓN DEL PROBLEMA

La recolección de basura es uno de los procesos relacionados con la gestión de residuos que se encarga de transportar los desechos producidos por las personas hacia los lugares de tratamiento de basura. Este proceso es uno de los más importantes relacionado con la salubridad pública en las ciudades modernas, debido a que evitan la acumulación de basura en las calles reduciendo la proliferación de enfermedades.

Según [Bošković *et al.*, 2016] el proceso de recolección de basura puede llegar a ser el responsable del 70 % del costo total del sistema de tratamiento de desechos, es por esto que es necesario encontrar rutas eficientes que permitan reducir el costo total del sistema cumpliendo con la recolección de todos los desechos producidos por las personas.

El problema abordado en esta memoria se centra en la recolección de residuos puerta a puerta con ventanas de tiempo. Este problema implica la obtención de una ruta con costo mínimo que comience en un vertedero, recorra todas las calles recolectando residuos de puerta a puerta y regrese al vertedero, que puede ser el mismo del inicio o diferente.

En este trabajo, se considera únicamente un vehículo de recolección de desechos con capacidad suficiente para recoger todos los residuos en la ruta sin necesidad de vaciar la carga durante el recorrido. Por lo tanto, no se toman en cuenta diferencias en la cantidad de basura en cada puerta. Además, el trabajo contempla la posibilidad de que, una vez que un camión realiza la recolección de residuos en una calle, pueda volver a pasar sin repetir la recolección, otorgándole al vehículo una bonificación de tiempo al evitar "detenerse" para realizar la recolección.

Este tipo de problemas se clasifica de manera general como un problema de enrutamiento de vehículos (ARP por sus siglas en inglés) y existen múltiples variantes que se categorizan principalmente por el tipo de arcos en el grafo, como:

- **Arcos no dirigidos:** Pueden ser visitados en cualquier sentido (*Undirected Graph*).
- **Arcos dirigidos:** Deben ser visitados en un solo sentido (*Directed Graph*).
- **Arcos mixtos:** Existen arcos que pueden ser visitados en cualquier sentido y otros en un solo sentido (*Mixed Graph*).
- **Arcos mixtos ventosos:** Similar al caso anterior pero los costos pueden variar dependiendo del sentido en el que se visitan (*Windy Graph*).

Particularmente, para el problema abordado en esta memoria, se utiliza un grafo con arcos mixtos. Esta condición resulta fundamental para evaluar el algoritmo propuesto en situacio-

nes del mundo real, ya que es esencial que el algoritmo respete la dirección de las calles para garantizar su efectividad.

Dentro de los grafos con arcos mixtos, el problema a estudiar lleva el nombre de *Mixed Rural Postman Problem* (MRPP en sus siglas en inglés) y consiste en determinar una ruta con costo mínimo que visite todos los arcos dirigidos o no dirigidos de un grafo mixto al menos una vez. Sin embargo, este problema será extendido para incorporar una nueva dimensión, la cual corresponde a las ventanas de tiempo para los arcos. En consecuencia, la ruta será penalizada por cada ventana de tiempo que no se respete.

2.1. Modelo Matemático

En esta sección se presenta el modelo matemático del problema a resolver, este modelo está basado en el modelo presentado en [Nobert y Picard, 1996] pero con la adición de restricciones para las ventanas de tiempo.

2.1.1. Parámetros

- N : Conjunto de nodos
- A : Conjunto de arcos dirigidos
- E : Conjunto de arcos no dirigidos.
- c_{ij} : Tiempo del arco dirigido (i, j) en el conjunto A
- d_{kl} : Tiempo del arco no dirigido (k, l) en el conjunto E
- L_{ij} : Margen inferior de la ventana de tiempo del arco (i, j)
- M_{ij} : Margen superior de la ventana de tiempo del arco (i, j)
- h_{ij}^k : Tiempo de partida desde el nodo i hacia el nodo j para la copia k
- V : Factor de reducción de tiempo

2.1.2. Notación

- S es un subconjunto de N
- $\bar{S} = N - S$
- $A^+(S) = \{(i, j) \in A : i \in S, j \in \bar{S}\}$: Conjunto de arcos dirigidos que salen de S

- $D^+(S) = |A^+(S)|$
- $A^-(S) = \{(i, j) \in A : i \in \bar{S}, j \in S\}$: Conjunto de arcos dirigidos que llegan a S
- $D^-(S) = |A^-(S)|$
- $E(S) = \{(i, j) \in E : (i \in S, j \in \bar{S}) \text{ o } (i \in \bar{S}, j \in S)\}$: Conjunto de arcos no dirigidos entre S y $\bar{S}$
- $D(S) = |E(S)|$
- $A(S) = A^+(S) \cup A^-(S)$: Conjunto de arcos que salen o llegan a S
- Cuando $S = \{i\}$ la notación anterior se escribirá como: $A_i^+, D_i^+, A_i^-, D_i^-, E_i, D_i$ y A_i
- $d_i = D_i^+ + D_i^- + D_i$: Grado del nodo i

2.1.3. Variables de decisión

- x_{ij} : Número de visitas del arco dirigido (i, j)
- y_{ij} : Número de visitas del arco no dirigidos (i, j)

2.1.4. Función Objetivo

La función objetivo presentada en la ecuación 1 indica que el número total de las copias que se agregan deben ser las menos posibles.

$$\text{Minimizar } Z = \sum_{(i,j) \in A} c_{ij}(1 + (x_{ij} - 1) \cdot V) + \sum_{(i,j) \in E} d_{ij}(1 + (y_{ij} - 1) \cdot V) \quad (1)$$

2.1.5. Restricciones

La ecuación 2 se relaciona con la paridad de los nodos del grafo. Esta ecuación es la encargada de añadir arcos al grafo con el fin de que el grado de todos los nodos sea par.

$$\sum_{(i,j) \in A_k} x_{ij} + \sum_{(i,j) \in E_k} y_{ij} = 2v_k + p_k, k \in N \quad (2)$$

La restricción presentada en la ecuación 3 esta diseñada para asegurarse de que el grafo sea balanceado. Esta ecuación, como señalan los autores, esta centrada en los arcos salientes de

un nodo ya que basta con solo esa restricción para asegurar el balance con los arcos de entrada.

$$(P) - \sum_{(i,j) \in A^+(S)} x_{ij} + \sum_{(i,j) \in A^-(S)} x_{ij} + \sum_{(i,j) \in E(S)} y_{ij} \geq u_S, S \subseteq N \quad (3)$$

Las restricciones en las ecuaciones 4 y 5 controlan el dominio de las variables de decisión del problema

$$x_{ij} > 0, (i, j) \in A \quad (4)$$

$$y_{ij} > 0, (i, j) \in E \quad (5)$$

$$v_i \geq 0, i \in N \quad (6)$$

Las ecuaciones 7, 8, 9 controlan que las ventanas de tiempo de los arcos del grafo se respeten.

$$L_{ij} \geq 0, (i, j) \in A \cup E \quad (7)$$

$$M_{ij} \geq 0, (i, j) \in A \cup E \quad (8)$$

$$L_{ij} \leq M_{ij}, (i, j) \in A \cup E \quad (9)$$

La ecuación 10 restringe que se respete el inicio de las ventanas de tiempo de cada arco

$$h_{ij}^0 \geq L_{ij}, (i, j) \in A \cup E \quad (10)$$

Las ecuaciones 11 y 12 controlan que el tiempo total de viaje de los arcos dirigidos y no dirigidos respectivamente no sea mayor al margen superior de las ventanas de tiempo de los arcos.

$$h_{ij}^k + c_{ij}^k \leq M_{ij}, \forall (i, j) \in A, \forall k \in [0, x_{ij}] \quad (11)$$

$$h_{ij}^k + d_{ij}^k \leq M_{ij}, \forall (i, j) \in E, \forall k \in [0, y_{ij}] \quad (12)$$

Las restricciones presentadas en 13 y 14 manejan que el factor de reducción de los tiempos de desplazamiento en los arcos sean válidos.

$$V > 0 \quad (13)$$

$$V \leq 1 \quad (14)$$

2.2. Resumen

En este capítulo, se presenta el problema de la recolección de desechos puerta a puerta, considerando la presencia de ventanas de tiempo. Este problema implica la tarea de diseñar rutas eficientes que cubran todas las calles de un sector específico, asegurando no solo la cobertura completa del área, sino también el respeto de las restricciones temporales y las orientación de las calles.

CAPÍTULO 3

ESTADO DEL ARTE

En este capítulo se presenta una revisión de la literatura existente para el problema del cartero chino y sus variantes más conocidas. La mayor parte de los artículos incluidos en este capítulo categorizan al problema de la recolección de desechos puerta a puerta como el problema del cartero chino o alguna de sus variantes o problemas similares.

Este capítulo está organizado en tres secciones. La primera sección describe el problema del cartero chino y sus principales variantes junto a problemas similares. Además, define y diferencia en profundidad las variantes más importantes en relación al problema estudiado en este trabajo de memoria. La segunda sección presenta técnicas de distintos autores para resolver el problema de recolección de desechos. Para finalizar, la última sección presenta procedimientos de múltiples autores para la resolución del problema de enrutamiento de arcos (*arc routing problem*) el cual se considera como la generalización del problema del cartero chino.

3.1. Problema del cartero chino

El problema del cartero chino (del inglés *chinese postman problem*) debe su nombre al matemático Kwan Mei-Ko que fue quien originalmente lo estudió en el año 1960. Este problema consiste en encontrar el camino cerrado más corto que permita visitar todos los arcos de un grafo no dirigido. Dicho de otra manera, se busca descubrir un camino que inicie y termine en el mismo nodo y que además visite todos los arcos de un grafo no dirigido buscando la ruta más corta.

Para este problema existen múltiples variantes y problemas similares que se diferencian principalmente en las características de los arcos presentes en los grafos y se pueden clasificar en las siguientes cuatro categorías:

1. Problemas con grafos no dirigidos
2. Problemas con grafos dirigidos
3. Problemas con grafos mixtos (poseen arcos dirigidos y no dirigidos)
4. Problemas con grafos ventosos

3.1.1. Problemas con grafos no dirigidos

En esta categoría se consideran los problemas donde *todos* los nodos del grafo tienen conexiones bidireccionales y el costo de recorrer en un sentido u otro es siempre el mismo. Los problemas más comunes en esta categoría se explican a continuación.

- Graphical TSP: El objetivo de este problema es visitar todos los nodos de un grafo iniciando y terminando en el mismo nodo. A diferencia del TSP, en este problema es posible volver a visitar un nodo ya recorrido [Carr y Simonetti, 2020].
- Rural postman problem: A diferencia del problema del cartero chino donde se tienen que recorrer todos los arcos de un grafo, en este problema solo es necesario recorrer ciertos arcos dejando a los demás como opcionales [Pearn y Wu, 1995].
- General routing problem: El problema acá consiste en recorrer un grafo con el objetivo de obtener un camino con el mínimo costo que visite ciertos nodos y arcos requeridos [Letchford y Eglese, 2001]. Usando este problema como base es posible obtener otros problemas modificando los parámetros originales, por ejemplo, si se eliminan los nodos requeridos para visitar es posible obtener el *rural postman problem*.
- Maximum benefit CPP: En este problema el objetivo es encontrar una ruta que maximice el beneficio de recorrer ciertos arcos de un grafo. Cada arco tiene asociado un costo y beneficio al ser atravesado. El costo y beneficio varían si al visitar o atravesar el arco este no se atiende o no se realiza el servicio [Pearn y Chiu, 2005].
- Length-Constrained k-Drones RPP: El problema consiste en recorrer todos los arcos de un grafo con una cantidad limitada de vehículos aéreos. A diferencia de los problemas anteriores, los vehículos en esta ocasión pueden recorrer completamente o no un arco, además de tener la libertad de cambiarse de arcos en cualquier punto y momento [Campbell *et al.*, 2021].

3.1.2. Problemas con grafos dirigidos

En la segunda categoría se encuentran problemas donde los nodos de un grafo están conectados a otros nodos con arcos dirigidos, eso quiere decir que solo se pueden recorrer en un sentido. A continuación se señalan algunos de los problemas más conocidos que se encuentran en esta categoría.

- Directed general routing problem: El problema consiste en encontrar una ruta con el costo mínimo que visite ciertos arcos y nodos requeridos de un grafo dirigido [Ávila *et al.*, 2015]. Este problema corresponde a una generalización por lo que a partir de este es posible obtener otras variantes, como por ejemplo el *stacker crane problem*.

- **Stacker crane problem:** A diferencia del problema anterior, el objetivo en este problema consiste en encontrar una ruta con el costo mínimo que empiece y termine en el mismo nodo y que visite todos los arcos de un grafo dirigido [Ávila *et al.*, 2015].
- **Close Enough Arc Routing Problem (CEARP):** Es un problema en el cual se debe encontrar una ruta con un costo mínimo que permita cubrir un conjunto de clientes que se ubican dentro o fuera de un grafo con vehículos que recorren los arcos del grafo y que pueden cubrir un radio r [Hà *et al.*, 2014].
- **Profitable CEARP:** Este problema es una extensión del CEARP en el cual cada cliente tiene un beneficio al ser atendido y el objetivo es obtener una ruta que maximice estos beneficios [Bianchessi *et al.*, 2022b].
- **Distance-constrained CEARP:** Similar al profitable CEARP, el objetivo de este problema es obtener múltiples rutas para recorrer un grafo en donde cada ruta no supere un largo o tiempo máximo [Corberán *et al.*, 2019].
- **Min-Max CEARP:** El objetivo de este problema consiste en encontrar distintas rutas para múltiples vehículos intentando mantener un balance entre las distancias de las rutas. Cada vehículo puede servir o recolectar los desechos de clientes desde una distancia si esta lo suficientemente cerca (de ahí el nombre del problema). [Bianchessi *et al.*, 2022a].
- **Team Orienteering ARP:** El problema tiene el objetivo de maximizar la ganancia total obtenida por una flota de vehículos que atienden a clientes regulares y potenciales [Archetti *et al.*, 2014].

3.1.3. Problemas con grafos mixtos

En la tercera clase de problema se tienen todos aquellos problemas que se basan en un grafo mixto. Este tipo de problemas pueden contener arcos dirigidos o no dirigidos (una combinación de las primeras dos categorías de problemas). Algunos de los problemas encontrados dentro de esta clase se describen a continuación.

- **Mixed postman problem:** Al igual que el CPP este problema consiste en encontrar una ruta que visite por lo menos una vez todos los arcos de un grafo, pero en este caso el grafo es mixto [Ralphs, 1993].
- **Mixed rural postman problem:** Consiste en encontrar la ruta con el costo mínimo que recorra un conjunto de arcos requeridos de un grafo mixto [Corberán *et al.*, 2000].
- **Mixed general routing problem:** Similar al GRP, este problema consiste en encontrar una ruta que recorra ciertos nodos y arcos en un grafo mixto, pero se diferencia del GRP en el cual este es un grafo mixto [Corberán *et al.*, 2005].

- Hierarchichal mixed rural postman problem: Este problema es una variante del *rural postman problem* en donde el grafo es mixto y ciertos arcos tienen mayor prioridad que otros [Colombi *et al.*, 2017]. Un ejemplo en el cual esta clase de problemas tiene aplicaciones en la vida real es en el *snow plowing* donde se remueve la nieve de las calles priorizando las calles principales.
- Periodic rural postman problem with irregular services on mixed graphs: Este problema es una extensión del RPP en donde se deben recorrer ciertos arcos una cantidad de veces en un periodo de tiempo. El objetivo de este problema es obtener múltiples rutas para cada periodo de tiempo en el cual se cumplan las restricciones de visitar los arcos una cantidad de veces. Este problema tiene aplicaciones practicas como por ejemplo en el mantenimiento de calles, en donde se deben servir ciertas calles cada cierto tiempo [Benavent *et al.*, 2019].

3.1.4. Problemas con grafos ventosos

La categoría de los grafos ventosos engloba a todos los problemas que tienen un grafo con arcos que tienen distintos costos dependiendo del sentido en el que se recorran. Dentro de la categoría se encuentran los problemas que se listan a continuación.

- Windy postman problem: Este problema es una variante del CPP en el que el costo de recorrer un arco depende del sentido en el que se visite [Guan, 1984]. Esta clase de problemas tiene aplicaciones cuando los caminos tienen condiciones que hacen asimétrico el recorrido, por ejemplo un camino en un cerro tendrá un costo superior si se sube en comparación a si se baja.
- Windy rural postman problem: Similar al problema anterior, en esta variante existen arcos que no es requisito que sean visitados [Benavent *et al.*, 2005].
- Windy general routing problem: Este problema surge de la combinación del GRP y la variante ventosa (*windy*). Consiste en la generalización de esta clase de problemas en el cual existe un subconjunto de los arcos y nodos que deben ser visitados. Además, cada arco tiene un costo de recorrido distinto dependiendo del sentido en el que se quiera recorrer [Corberán *et al.*, 2007].
- Min-Max K-vehicles windy RPP: De forma parecida a los problemas anteriores, esta variante consiste en la combinación de múltiples variantes, como el *Min-Max RPP* y el *Windy RPP* en donde el objetivo es encontrar un conjunto de rutas balanceadas en el que cada arco tiene un costo que depende del sentido en el que se recorra [Benavent *et al.*, 2014].
- Profitable windy RPP: Al igual que el *profitable RPP*, este problema consiste en encontrar una ruta que maximice el beneficio obtenido pero se diferencia en que el beneficio y/o costo depende del sentido en el que se visiten los arcos [Ávila *et al.*, 2016].

La Tabla 1 presenta un listado de las principales variantes del problema estudiado. En cada caso, se detalla el tipo de arcos empleados (ya sean no dirigidos, dirigidos o de naturaleza mixta), los arcos que requieren obligatoria visita (si aplica), y los nodos que también requieren obligatoria visita (si aplica).

Tipos de Arco	Problema	Arcos a visitar	Nodos a visitar
No dirigidos	Graphical TSP	N/A	Todos
	RPP	Algunos	N/A
	GRP	Algunos	Algunos
	Maximum benefit CPP	Algunos	N/A
	Length-Constrained k-Drones RPP	Todos	N/A
Dirigidos	DGRP	Algunos	Algunos
	Stacker Crane Problem	Todos	N/A
	CEARP	Algunos	N/A
	Distance-constrained CEARP	Algunos	N/A
	Min-Max CEARP	Algunos	N/A
Mixtos	Team Orienteering ARP	Algunos	N/A
	Mixed Postman Problem	Todos	N/A
	Mixed RPP	Algunos	N/A
	Mixed GRP	Algunos	Algunos
	Hierarchichal Mixed RPP	Algunos	N/A
	Periodic rural postman problem with irregular services on mixed graphs	Algunos	N/A

Tabla 1: Resumen de las variantes

3.1.5. Combinación de variantes

En la literatura no solo existen las variantes señaladas anteriormente, sino que además se pueden encontrar versiones que se componen de una combinación de variantes del problema original.

El problema mixto del cartero rural (*mixed rural postmap problem*) [Corberán *et al.*, 2000] consiste en la combinación del problema mixto del cartero chino junto al problema del cartero rural, por lo que el objetivo de este problema implica encontrar una ruta que minimice los costos de recorrer ciertos arcos obligatorios de un grafo. Además este grafo presenta arcos que solo pueden ser recorridos en un sentido y arcos que no son necesarios de cubrir.

En el problema del cartero ventoso rural (*windy rural postman problem*) [Benavent *et al.*, 2005] se considera la combinación del problema del cartero rural con el problema del cartero ventoso y se caracteriza en buscar la ruta que visite ciertos arcos requeridos y donde, además, el costo de algunos arcos varía dependiendo del sentido en el que se recorran.

3.2. Problemas de recolección de desechos

Los problemas de recolección de desechos han estado presentes a lo largo de la historia moderna, pero han obtenido mayor notoriedad en los últimos años debido al aumento de

la población en las ciudades. Debido a esto, en esta sección se señalan algunos enfoques que han sido estudiados en tiempos más recientes con el fin de encontrar soluciones más eficientes al problema de la recolección de desechos.

En [Corberán *et al.*, 2002], los autores plantean una solución basada en la técnica *Greedy Randomized Adaptive Search Procedure* (GRASP) para solucionar el problema mixto del cartero chino [Feo y Resende, 1995]. Esta técnica consiste en iteraciones constructivas a partir de una solución pseudo-aleatoria que luego es mejorada iterativamente a través de una búsqueda local. Durante la fase constructiva el algoritmo se encarga de generar una solución inicial usando una función *greedy* que evalúa la conveniencia al seleccionar la orientación en la que se recorren los arcos. La función *greedy* empieza con los arcos que no tengan una dirección establecida y termina cuando todos los arcos hayan sido evaluados.

La segunda parte del algoritmo se encarga del mejoramiento de la solución. A partir de la solución construida en la primera parte, la fase de mejoramiento, busca encontrar la mejor solución óptima localmente. Este proceso se encarga de realizar una serie de movimientos o intercambios en los pares de vértices en los caminos que se recorren para luego calcular si los cambios realizados mejoran o empeoran la solución inicial hasta que el algoritmo no pueda seguir mejorando la solución.

Esta estrategia fue probada con instancias de pruebas generadas aleatoriamente con hasta 200 nodos y 600 arcos. Además, a partir de estas instancias de prueba, se realizaron múltiples evaluaciones variando la cantidad de arcos dirigidos en el grafo. Los resultados obtenidos usando esta estrategia son cercanos al óptimo con una desviación del 1,12 % en todas las instancias con las que se probó el algoritmo.

En [Groves y Vuuren, 2005] los autores proponen un método para buscar soluciones al problema del cartero rural basado en heurísticas que se han utilizado en problemas de enrutamiento de vehículos (*vehicle routing problem*) y en el problema del vendedor viajante (*traveling salesman problem*). El método propuesto consiste en aplicar movimientos a una solución inicial generada usando la heurística de Frederickson [Frederickson *et al.*, 1976]. Esto, con el objetivo de modificar el orden y sentido con el que se visitan ciertos arcos. Este método se para el movimiento en dos partes, la primera parte corresponde al cambio de orden de visita de los arcos y la segunda a la determinación del sentido de visita de los arcos. En la primera parte del algoritmo se eligen dos o tres arcos, usando 2-opt o 3-opt, de los movimientos posibles para luego eliminarlos con el fin de encontrar (o generar) otros arcos que generen un camino más corto. Este proceso se repite hasta que la solución no pueda ser mejorada. La figura 1 muestra un ejemplo de la aplicación del movimiento donde se eliminaron dos arcos de la ruta (los tachados en la figura 1) y se generó una nueva usando los arcos punteados.

En la segunda parte del algoritmo, para determinar el sentido de visita de los nodos, se realiza una comprobación verificando todas las posibles combinaciones en las que un nodo pueda ser visitado hasta encontrar la ruta con el costo más bajo.

El algoritmo propuesto por los autores fue probado con múltiples instancias. Los grafos más

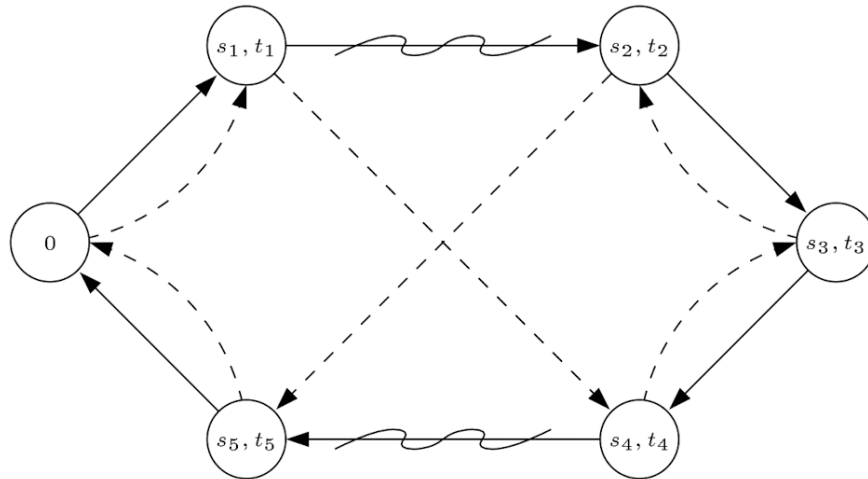


Figura 1: Generación de una nueva ruta luego de eliminar 2 arcos

Fuente: [Groves y Vuuren, 2005].

grandes donde se aplicó el método contenían un poco más de 3.000 nodos y una cantidad entre 20.000 a 28.000 arcos. Los resultados indican que, en general, el algoritmo desarrollado encuentra la solución óptima o una mejor en comparación al algoritmo de *Frederickson* [Frederickson *et al.*, 1976] el cual era el mejor hasta la fecha de publicación del artículo. Sin embargo, estos resultados necesitaron un tiempo de cómputo más elevado (entre un 9 % a un 35 %). Los autores concluyen que esta propuesta no solo puede servir para el problema del cartero rural si no que para cualquier problema de enrutamiento de arcos.

A diferencia de los autores anteriores, [Li y Huang, 2010] plantean una nueva estrategia basada en programación probabilística debido a que consideran el problema como uno no determinista. Los autores indican que en la gestión de la recolección de basura existen incertidumbres en múltiples parámetros los cuales afectan la toma de decisiones. Algunos de estos parámetros son:

- Tasa de generación de basura
- Flujo de los residuos a las plantas de tratamiento o vertederos.
- Costos de operación de las plantas de tratamiento o vertederos.
- Costo de penalización por tratar el exceso de residuos
- Ingresos por el tratamiento de residuos
- Entre otros.

Los autores aplican técnicas probabilísticas para reducir la incertidumbre al trabajar con parámetros reales generando planificaciones de rutas más eficientes capaces de abordar las

Nivel de generación	Probabilidad de generación
[1373, 1433]	0,125
[1434, 1494]	0,280
[1495, 1577]	0,404
[1578, 1648]	0,191

Tabla 2: Tasa de generación de basura en el periodo 1

Nivel de generación	Probabilidad de generación
[1418, 1510]	0,193
[1511, 1605]	0,575
[1606, 1702]	0,232

Tabla 3: Tasa de generación de basura en el periodo 2

características dinámicas de problemas reales. Los autores obtienen múltiples soluciones para el problema de la asignación de flujo de residuos. Cada una de las soluciones expresa una combinación de rangos de los parámetros del problema. Por ejemplo el primer caso considera todos los parámetros en su valor más bajo mientras que el último caso contiene los valores más altos que pueden tomar los parámetros. Estas soluciones permiten expresar los distintos escenarios que se pueden encontrar con la variación de la generación de basura.

Las instancias de prueba se dividían en tres periodos de tiempo donde cada periodo tiene una tasa de generación de basura distinta. La tasa para los periodos 1, 2 y 3 se pueden observar en las tablas 2, 3 y 4 respectivamente. Además de la tasa de generación, cada periodo cuenta con un flujo o destino de los residuos, los cuales se encuentran en la tabla 5.

En [Braier *et al.*, 2017] los autores propusieron una técnica basada en programación lineal entera (*Integer Linear Programming* en inglés) para encontrar rutas optimizadas para los vehículos de recolección de desechos reciclables que pasen puerta a puerta por provincias de Argentina. El modelo matemático propuesto por los autores optimiza el costo total de las rutas de recolección de desechos asegurando que todas las calles sean recolectadas de manera efectiva, sin dejar ninguna sin atender. Además, este modelo genera rutas con menor consumo de tiempo y energía para la recolección de residuos, como efectos secundarios. El modelo además incluye las restricciones de tráfico presentes en las calles, como la prohibición de doblar a la izquierda en calles de doble sentido en un cruce con semáforos.

La solución propuesta por estos autores fue probada con casos de prueba obtenidos direc-

Nivel de generación	Probabilidad de generación
[1465, 1555]	0,185
[1556, 1648]	0,605
[1649, 1759]	0,210

Tabla 4: Tasa de generación de basura en el periodo 3

Destino	Periodo 1	Periodo 2	Periodo 3
Vertedero	[700, 950]	[600, 850]	[500, 750]
Compostaje	[100, 200]	[250, 300]	[300, 400]
Reciclaje	[200, 300]	[300, 400]	[350, 450]

Tabla 5: Flujo de basura

tamente de la municipalidad de la provincia. Estos casos contienen entre 304 y 1254 nodos, 12 a 608 arcos que pueden ser recorridos en cualquier sentido y entre 17 a 181 arcos que deben ser visitados en un único sentido. Tal como se señaló anteriormente, los resultados muestran que, en cada caso de prueba, se logró obtener por lo menos una ruta que recorre el 100 % de la localidad con una distancia muy similar a las rutas que se tenían antes. Además, en ciertas instancias se logró reducir la distancia total que debían recorrer los vehículos. Es importante señalar que los autores limitaron a 10 minutos el tiempo de ejecución para generar las soluciones de los casos de prueba, por lo que existen ciertas instancias que no lograron el óptimo. Sin embargo, todos los casos de prueba difieren en menos de un 1 % del valor óptimo.

Los autores señalan que su solución propuesta genera rutas más eficientes y eran capaces de recorrer un 100 % de la provincia. Esto en comparación a las rutas que seguían los camiones anteriormente, las cuales eran construidas manualmente.

En Renca, una comuna de Chile, los autores [Blazquez y Paredes-Belmar, 2020] diseñaron un sistema de recolección de basura que, a diferencia de lo usado en Argentina por [Braier *et al.*, 2017], no recorre puerta por puerta toda una comuna. La solución propuesta por estos autores se basa en que las personas deban transportar sus desechos a contenedores ubicados en lugares estratégicos. Luego, dichos contenedores, deben ser recolectados por los camiones de recolección de basura. La obtención de las ubicaciones de los contenedores dentro de la comuna estudiada por los autores se basó en un modelo de programación entera mixta. Para esto se consideraron dos tipos de contenedores, uno pequeño y otro grande. Al igual que para las ubicaciones de los contenedores, la obtención de rutas se obtuvo usando un modelo de programación lineal entera mixta, junto a una técnica de búsqueda de grandes vecindarios *large neighborhood search* [Shaw, 1998].

La estrategia de búsqueda en grandes vecindarios es una técnica de búsqueda heurística que consiste en cambiar múltiples características de una solución para encontrar un óptimo en un gran vecindario de soluciones, todo esto con el objetivo de escapar de mínimos locales. Al ser una técnica de búsqueda local es necesario tener una solución inicial la cual es obtenida por los autores usando un método *greedy* para luego ser mejorada utilizando una estrategia de destruir y reparar la cual consiste en eliminar nodos de la solución y después reparar la solución usando una técnica *greedy*.

3.3. Problema de enrutamiento de arcos

Los problemas de enrutamiento de arcos son un problema fundamental en la optimización de rutas en diversas aplicaciones, desde la logística de transporte y distribución, hasta la gestión de recolección de desechos. Estos problemas consisten en encontrar la ruta óptima para uno o más vehículos que deban visitar un conjunto de ubicaciones recorriendo los arcos que las conectan con el objetivo de minimizar la función de costo la cual puede incluir múltiples factores como distancia, tiempo, consumo de combustible, capacidad, entre otras.

En esta sección, se exhibirán algunos trabajos que han logrado solucionar alguna de las variantes del problema que han sido mencionadas previamente.

Los autores [Gordenko y Avdoshin, 2018] señalan que la gran parte de los problemas de enrutamiento pueden ser convertidos o transformados en otros problemas del mismo tipo. Es por esto que los autores proponen transformar los problemas de tipo Arc Routing Problems ARP (alguna variantes del *Chinese Postman Problem*) en problemas de tipo *Generalized Traveling Salesman Problem (GTSP)*.

La ventaja de transformar los ARP a GTSP es que estos últimos en verdad son *Node Routing Problem* por lo que es posible aplicar cualquier técnica conocida que permita solucionar el problema del vendedor viajero. Uno de los problemas más estudiados en la literatura.

Siguiendo con esta idea, los autores [Pearn *et al.*, 1987] proponen un algoritmo para transformar el problema del cartero chino capacitado (*Capacitated Chinese Postman Problem*) a un problema de enrutamiento de nodos *Node Routing Problem*. Esta conversión se realiza transformando los arcos del grafo original en tres nodos conectados equidistantes sobre el arco, tal como se aprecia en la figura 2.

Si bien los autores señalan que esta estrategia permite aplicar algoritmos utilizados en el enrutamiento de nodos para solucionar problemas del tipo ARP, indican que es necesario realizar más estudios para determinar si es eficiente computacionalmente. Esto, debido a que la cantidad de nodos aumenta significativamente dependiendo de la instancia a resolver, llegando a triplicar este valor en ciertas instancias.

A diferencia de las estrategias nombradas anteriormente, los autores de [Hertz *et al.*, 2000] proponen usar *Tabu Search* para solucionar el *Capacitated Arc Routing Problem*. Esta técnica es un método de búsqueda local que consiste en indagar el vecindario de una solución con el objetivo de encontrar una solución mejor a la actual. Este método evita quedar estancado en un óptimo local aceptando soluciones de peor calidad y utilizando una lista (lista tabú) que bloquea ciertos movimientos con el fin de prohibir volver a soluciones ya encontradas.

Debido a que la técnica planteada es un método incompleto, es necesario obtener una solución inicial para poder aplicar el algoritmo. Los autores señalan que la heurística para encontrar una solución inicial que utilizan corresponde a lo planteado por [Frederickson, 1979] en 1979 el cual describe un algoritmo heurístico $O(n^3)$ que encuentra una solución usando un

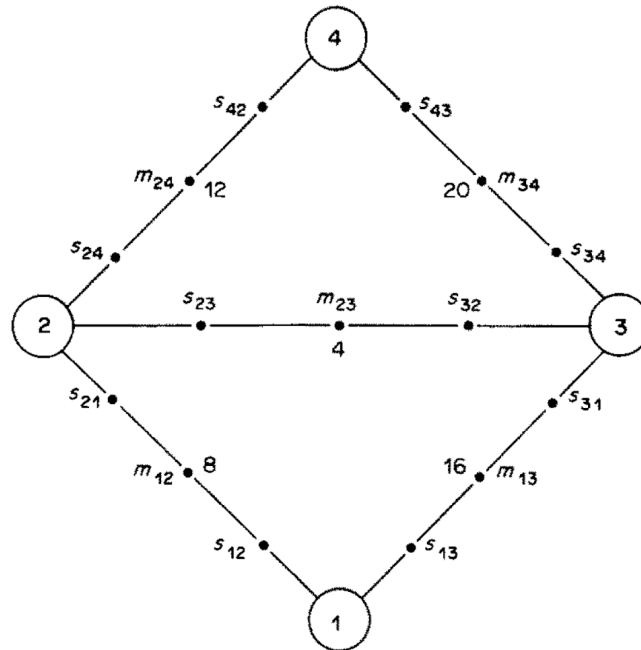


Figura 2: Ejemplo de enrutamiento de arcos con la incorporación de nuevos nodos (los costos originales de los arcos se muestran junto a cada uno de ellos). Fuente: [Pearn et al., 1987]

árbol de expansión mínima conectando los nodos de la forma más eficiente que se pueda.

Los autores probaron su algoritmo con dos conjuntos de pruebas. El primero contenía 23 instancias de pruebas las cuales tenían entre 7 y 27 vértices y 11 a 55 arcos y todos eran requeridos. El segundo conjunto contiene 34 instancias donde los vértices varían entre 24 a 50 y los arcos entre 34 y 97, nuevamente todos son requeridos.

En el primer conjunto de pruebas, los resultados obtenidos por los autores señalan que de las 23 instancias de prueba se generaron soluciones óptimas en 18 instancias y se encontró una mejor solución en 20 instancias. Además, los autores señalan que los resultados se alejan en promedio un 0,17 % de la mejor solución encontrada hasta la fecha y esta diferencia jamás supera el 1,93 %.

En el segundo conjunto, el algoritmo logró obtener en 26 instancias el mejor valor conocido junto con los algoritmos con los que se comparaba y en seis casos obtuvo el mejor valor mientras que los demás no lo lograron. Además, los valores obtenidos señalan una desviación en promedio del 1,13 % del mejor valor para cada instancia y en el peor de los casos un 8,10 %.

Junto a las dos pruebas anteriores, los autores también evaluaron su propuesta en 270 instancias más grandes con un rango de vértices entre 20, 40 y 60 y donde los arcos eran generados aleatoriamente usando un ciclo hamiltoniano. Además se añadían arcos al azar uti-

lizando densidades que variaban entre $[0,1, 0,3]$, $[0,4, 0,6]$ y $[0,7, 0,9]$ y de estos arcos se seleccionaban cuáles iban a ser requeridos utilizando las mismas densidades respectivas.

En comparación al problema abordado en esta memoria, el *Capacitated Arc Routing Problem* (CARP) se diferencia principalmente en que cada arco tiene una demanda (positiva) la cual debe ser satisfecha, los vehículos que recorrerán la ruta parten y terminan en el mismo lugar y estos tienen una capacidad limitada. Todas estas suposiciones se omiten en este proyecto de memoria.

Similar al planteamiento de los autores anteriores, en [Babaei Tirkolaee *et al.*, 2016] enfrentan el mismo problema del CARP usando un algoritmo basado en *Simulated Annealing* para el desafío de la recolección de basura puerta a puerta. Ambas técnicas (SA y TS) al ser incompletas requieren de una solución inicial, en este caso los autores proponen un algoritmo heurístico capaz de crear rutas iniciales para cada vehículo para luego mejorarlas usando SA.

Los pasos del algoritmo heurístico para generar las soluciones iniciales son los siguientes:

1. Empezar con el primer vehículo.
2. Seleccionar una cantidad k de arcos requeridos que tengan la menor distancia del nodo seleccionado (o nodo inicial si es la primera iteración).
3. Escoger un arco aleatoriamente de los arcos seleccionados en el paso anterior y luego realizar el servicio.
4. Si se han visitado todos los arcos requeridos el algoritmo termina y retorna la ruta. En cualquier otro caso se vuelve al paso 2.

Los movimientos utilizados por el SA para mejorar las soluciones iniciales consisten en:

1. Se seleccionan dos recorridos aleatoriamente y si tienen uno o más bordes mutuos entre los recorridos se selecciona aleatoriamente uno para luego dividir en 2 partes los recorridos generando dos nuevos caminos.
2. Se seleccionan dos recorridos aleatoriamente y si tienen dos bordes mutuos entre ellos se cambia la secuencia entre los dos bordes mutuos
3. Un arco de una ruta es seleccionada aleatoriamente y se invierte su dirección.
4. Una parte de una ruta se selecciona aleatoriamente y se cambian las direcciones.

Los autores evaluaron el algoritmo en diez instancias de prueba las cuales tenían entre 8 a 90 arcos y de esos entre 6 a 72 eran requeridos. Los resultados señalan que el algoritmo

propuesto es eficiente y que obtiene en gran parte de las instancias un valor cercano al óptimo requiriendo menor tiempo de cómputo en comparación a programas de optimización como CPLEX. Además, los resultados obtenidos por el algoritmo propuesto y CPLEX no se diferenciaban en más de 3 %.

Los autores [Lacomme *et al.*, 2001] proponen un algoritmo genético para solucionar el problema CARP. El algoritmo inicia con tres buenas soluciones generados por distintas heurísticas y las demás son soluciones pseudo-aleatorias. Cada solución se representa como una secuencia de tareas las cuales indican el arco y el orden en el que se visitan. Al generar las soluciones, el algoritmo prohíbe las soluciones iguales para poder generar un mayor número de variación en la población.

Las tres buenas soluciones a las que hacen referencia los autores son generadas utilizando la heurística de Ulusoy's, *Path-Scanning* y *Augment-Merge*. La primera heurística obtiene las soluciones generando un recorrido gigante que cubra todos los nodos para luego separar el recorrido en múltiples caminos que puedan ser visitados respetando las capacidades de los vehículos. La segunda heurística genera las soluciones comenzando desde un depósito y extendiendo el camino nodo por nodo dependiendo si los nodos cercanos tienen que ser atendidos o no. La tercera heurística genera las soluciones construyendo los caminos seleccionando los nodos uno a uno en un orden decreciente respecto a al costo. Para más detalles sobre las heurísticas de las soluciones revisar [Lacomme *et al.*, 2001].

La reproducción en el algoritmo genético se realiza seleccionando al azar dos soluciones de la población para luego elegir la que tenga menor costo para ser el primer padre. Luego este proceso se repite para obtener el segundo padre para finalizar con un hijo aplicando *order crossover* para obtener la descendencia. El *order crossover* es un operador utilizado para combinar dos cromosomas en algoritmos genéticos en donde se selecciona una parte de los cromosomas y es intercambiada entre los padres. Una vez se tenga un hijo con los genes de los padres, se aplica un operador de mutación que usa búsqueda local para mejorar la solución. Este operador se encarga principalmente de verificar todas las permutaciones de dos tareas que puedan realizarse en una ruta.

Los autores evaluaron el algoritmo genético en dos conjuntos de prueba. El primero contiene 23 instancias de pruebas con entre 7 a 27 nodos y 11 a 55 arcos. El segundo conjunto contiene 34 instancias de prueba más complejas que tenían entre 24 a 50 nodos y 34 a 97 arcos. Los resultados para el primer conjunto de prueba señalan que el algoritmo genético es eficiente en todos los casos de prueba, siendo similar al presentado por [Hertz *et al.*, 2000] y otros con los que se comparó e incluso mejorando en dos instancias el mejor resultado hasta la fecha. Para el segundo conjunto de pruebas, el algoritmo logra obtener cinco soluciones mejores y logra superar en 16 ocasiones al mejor algoritmo el cual corresponde al desarrollado por [Hertz *et al.*, 2000] hasta la fecha de publicación del artículo. En ambos casos, el promedio de desviación entre los resultados del algoritmo y la mejor solución es de un 0,2 % y 1,1 % para el primer y segundo conjunto de pruebas respectivamente.

El CARP puede modelarse como un problema de programación lineal entera por lo que au-

tores como [Belenguer y Benavent, 2003] han propuesto algoritmos del tipo *Cutting Plane Method* para solucionar este problema. Este tipo de método de resolución se basa en ir limitando las soluciones usando las restricciones del problema modeladas como ecuaciones matemáticas a diferencia de los métodos anteriores que van construyendo y/o mejorando la solución.

El algoritmo propuesto fue evaluado con cuatro conjuntos de prueba. El primer conjunto tiene instancias con una cantidad de nodos que variaba entre 24 a 50 y entre 34 a 97 arcos. El segundo conjunto contiene instancias con una cantidad de arcos entre 11 a 55 y entre 7 a 27 nodos. El tercer conjunto tiene entre 6 a 10 vértices y 5 arcos requeridos. El último conjunto tiene instancias que contienen entre 77 a 140 nodos y una cantidad de arcos requeridos que varía entre 51 y 190.

Los resultados que se obtienen luego de aplicar este método de resolución reflejan que este procedimiento es bastante eficiente, obteniendo resultados de calidad en un tiempo de cómputo bajo en las instancias más grandes que se probaron. La desviación en promedio con las mejores soluciones conocidas para los conjuntos de prueba son 0,41 %, 0,14 %, 0,58 % y 2,4 % para los conjuntos 1, 2, 3 y 4 respectivamente.

Además, se señala que en una gran cantidad de casos de prueba el algoritmo encontró el resultado óptimo, es decir, el límite inferior coincidía con el límite superior. Sin embargo, los autores señalan que hay ciertas instancias que no se tienen los resultados óptimos por lo que en esos casos no se pudo comparar y obtener el porcentaje de diferencia.

3.4. Resumen

En este capítulo se presentó el problema del cartero chino y se analizó las diferentes variantes que existen de este problema en la literatura. Se categorizaron las variantes en cuatro grupos dependiendo del tipo de grafo en el que el modelo del problema es planteado: grafos dirigidos, grafos no dirigidos, grafos mixtos y grafos ventosos. Junto con lo anterior, también se examinaron múltiples combinaciones de variantes las cuales describen situaciones más cercanas a la realidad y con aplicaciones prácticas en la vida real.

Por otro lado, se abordaron problemas de recolección de desechos y los métodos utilizados por distintos autores para resolver esta clase de problema a lo largo del tiempo. Incluyendo estudios de recolección puerta a puerta y por contenedores.

Por último se presentó el planteamiento de distintos autores para resolver el problema de enrutamiento de arcos el cual es la generalización del problema del cartero chino. Se presentan diferentes aplicaciones de técnicas completas y heurísticas para problemas de tamaños acotados en cantidad de nodos y arcos.

CAPÍTULO 4

PROPUESTA DE SOLUCIÓN

En este capítulo, se introduce un algoritmo heurístico para la resolución del problema de ruteo de vehículos de recolección de residuos puerta a puerta con ventanas de tiempo. El algoritmo propuesto se basa en la optimización por colonias de hormigas, que ha demostrado ser una estrategia metaheurística exitosa en una amplia variedad de problemas de optimización combinatoria.

La utilización de colonias de hormigas para abordar problemas de optimización tiene sus orígenes en 1996, cuando los autores [Dorigo *et al.*, 1996] presentaron un modelo que empleaba analogías con las colonias de hormigas naturales para resolver esta clase de problemas.

El modelo que presentaron destacaba por su retroalimentación positiva, que se lograba a través del uso de un mecanismo de comunicación centralizado denominado matriz de feromonas y por el cálculo distribuido de las soluciones utilizando hormigas artificiales. Dichas hormigas artificiales permiten obtener soluciones de buena calidad en tiempos de cómputo acotados.

4.1. Representación de las soluciones

Para representar una solución del problema, se utiliza una lista en la que cada elemento consiste en un movimiento que realiza una hormiga en el grafo del problema. Cada movimiento contiene un nodo inicial, un nodo final, el tiempo que tarda en realizarse y la penalización correspondiente (en caso de haberla) por realizar el movimiento.

Dado que el algoritmo propuesto trabaja con soluciones infactibles respecto a las ventanas de tiempo, es necesario tener una representación que permita distinguir entre soluciones factibles e infactibles. Por lo tanto, la representación utilizada permite determinar si una solución es factible o no mediante la suma de las penalizaciones correspondientes a cada movimiento en la solución. Si la suma total de penalizaciones es 0, la solución propuesta es factible; de lo contrario, no lo es.

En la figura 3 se puede apreciar un esquema de la representación propuesta.

Inicio: A Final: B Tiempo de viaje: 2 Penalización: 1	Inicio: B Final: C Tiempo de viaje: 1 Penalización: 3	Inicio: C Final: D Tiempo de viaje: 3 Penalización: 0	Inicio: D Final: E Tiempo de viaje: 2 Penalización: 4
--	--	--	--

Figura 3: Ejemplo de una solución utilizando la representación propuesta

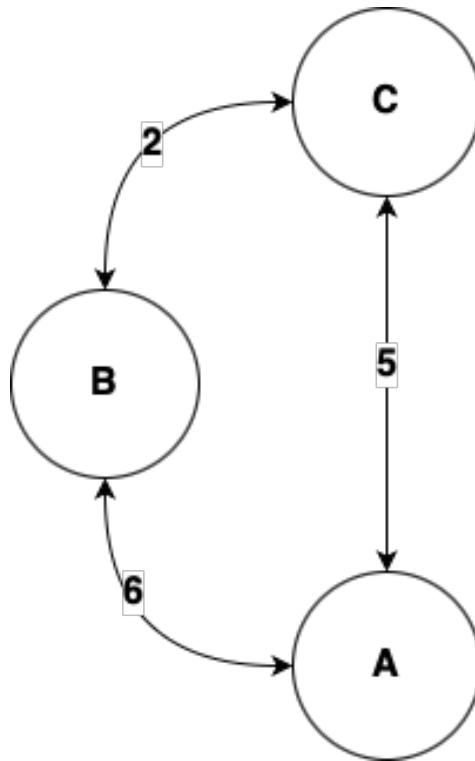


Figura 4: Ejemplo de una instancia pequeña para el problema del ruteo de vehículos de recolección de residuos puerta a puerta con ventanas de tiempo. Fuente: Propia

En la figura 6 se observa un esquema detallado de la solución obtenida para la instancia de prueba mostrada en la figura 4. Esta instancia se representa mediante un grafo no dirigido, donde cada arco indica los tiempos de viaje necesarios para realizar la recolección de desechos. En este caso, el objetivo es recorrer todos los arcos, iniciando desde el nodo A y terminando en el nodo C, minimizando el tiempo total de viaje.

Para esta instancia de prueba, el algoritmo propuesto encontró la solución que se indica en la figura 6. La solución comienza en el nodo A y finaliza en el nodo C con un tiempo total de viaje de 18 unidades de tiempo.

Es importante destacar que la solución obtenida para la instancia de prueba fue obtenida con los siguientes supuestos y parámetros:

- Las ventanas de tiempo son lo suficientemente grandes para no incurrir en penalizaciones.
- El tiempo de viaje para un arco ya visitado no es reducido.

Ambos supuestos fueron establecidos con el fin de facilitar el entendimiento de la representación de la solución y son ajustables para cada instancia. Por ejemplo, si para la instancia

Inicio: A Final: B Tiempo de viaje: 6 Penalización: 0	Inicio: B Final: C Tiempo de viaje: 2 Penalización: 0	Inicio: C Final: A Tiempo de viaje: 5 Penalización: 0	Inicio: A Final: C Tiempo de viaje: 5 Penalización: 0
--	--	--	--

Tabla 6: Representación de la solución de la instancia de ejemplo Fuente: Propia

de la figura 4 se establece un porcentaje de reducción del tiempo de viaje para los arcos ya visitados de un 80 %, el tiempo total de viaje sería de 14 unidades de tiempo siguiendo el mismo camino.

4.2. Función de evaluación

La función de evaluación es utilizada por las hormigas para evaluar la calidad de las soluciones generadas y se presenta en la ecuación 15. En dicha ecuación, T_e representa la suma de los tiempos de viaje del arco e , mientras que P_e indica la acumulación de penalizaciones cada vez que el arco e es visitado fuera de su ventana de tiempo. Además, como el objetivo es visitar todos los arcos del problema, estos son representados por E .

$$\text{MIN } T = \sum_e^E T_e + P_e \quad (15)$$

4.2.1. Tiempos de viaje

El cálculo del tiempo de viaje de una ruta en el algoritmo depende del número de veces que cada arco es visitado. En general, se asume que una vez que un arco ha sido visitado, el tiempo de visitarlo nuevamente, las veces que sea, será menor. Esto se debe a que, en el problema de recolección de basura, una vez que una calle ha sido recolectada, se asume que las siguientes veces que un camión pase por dicha calle será realizando un recorrido y no recolectando.

Para considerar esta situación, se utiliza una función que aplica un factor que reduce el tiempo de viaje de cada arco en relación al tiempo base cuando se visita el arco más de una vez.

En la expresión 16, se observa el cálculo del tiempo total de viajes del arco e (T_e). En dicha expresión, x_e representa las veces que el arco e ha sido recorrido, t_e es el tiempo base de viaje, y v es el factor de reducción del tiempo de viaje por cada recorrido previo del arco.

Por ejemplo, si un arco se visita una vez, el tiempo total de viaje (T_e) será igual al tiempo base de viaje del arco. En cambio, si un arco ha sido visitado más de una vez, el tiempo total

se calculará utilizando el tiempo base de viaje del arco, sumado a $x_e - 1$ veces el tiempo de viaje base del arco reducido por el factor v , tal como se aprecia en la expresión 16

$$T_e = t_e(1 + (x_e - 1) \cdot v) \quad (16)$$

4.2.2. Penalización por ventana de tiempo

La función de penalización por ventana de tiempo tiene como objetivo castigar a las rutas que recorren los arcos fuera de su rango de tiempo. Esta función se define como la diferencia entre el tiempo en el que se inicia o termina el desplazamiento (o recolección) de un arco con el límite inferior o superior de la ventana de tiempo del arco, dependiendo si se visita el arco antes de la ventana de tiempo o después respectivamente.

Esta función retorna 0 cuando los arcos se visitan dentro del margen de su ventana de tiempo. En caso contrario, la penalización es proporcional a la ventana de tiempo y los tiempo de llegada o salida del arco seleccionado, aumentando a medida que se visite un nodo tardíamente o anticipadamente. Es decir, cuanto más se aleje la hora de llegada o salida de las horas límites de la ventana de tiempo, mayor será la penalización.

En la ecuación 17 se observa el cálculo para obtener la penalización de un nodo al visitarlo en el tiempo t , en donde A y B corresponden al límite inferior y superior de su ventana de tiempo, respectivamente y t_e indica el tiempo de viaje asociado al arco e .

$$p_e = \max(A - t, 0) + \max((t + t_e) - B, 0) \quad (17)$$

Consideremos la instancia ilustrada en la figura 5 en donde la ventana de tiempo del arco AB es de $[6, 10]$ unidades de tiempo y el tiempo de viaje del mismo arco tiene un valor de 3 unidades de tiempo. Supongamos que se visita el arco AB en $t = 0$, en este caso, la penalización asignada a este movimiento tendrá un valor de 6 unidades de tiempo debido a que se visita el arco antes de su ventana de tiempo. Similarmente, en caso de visitar el mismo arco en $t = 12$, la penalización de este movimiento será de 5 por la diferencia con el límite superior de su ventana de tiempo. Y por último, si se visita el arco en $t = 9$, la penalización será de solo 2 unidades de tiempo debido a que la recolección termina 2 unidades de tiempo fuera de la ventana de tiempo del arco.

Es importante señalar que la penalización se realiza para todos los desplazamientos realizados por las hormigas, se realice una colección de basura o no.

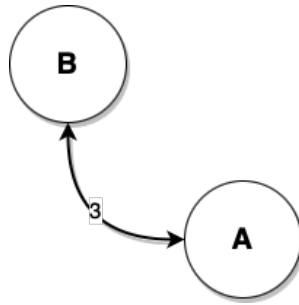


Figura 5: Grafo de una instancia con 2 nodos y un arco. Fuente: Propia

4.3. Estructura

La propuesta se estructura como se muestra en el algoritmo 1. El algoritmo corresponde a un algoritmo tipo *ant system* y se divide en tres partes fundamentales.

La primera parte es la inicialización (línea 1), donde se crea la matriz de feromonas y se asigna un valor inicial de feromonas a cada posible arco de la instancia que se desea resolver.

La segunda parte se centra en la construcción de soluciones, que corresponde al núcleo del algoritmo. En esta parte es donde se generan las soluciones utilizando múltiples parámetros, que se describen más adelante. Esta sección corresponde al código entre las líneas 2 y 11.

La última es la actualización de feromonas (línea 12), un proceso fundamental que permite la exploración y explotación del algoritmo. En este proceso, las feromonas se evaporan y/o se depositan en la matriz para fomentar la exploración y la explotación respectivamente.

Algorithm 1 Algoritmo Propuesto

```

1: inicializar()
2: mejor_solucion ← Solución Vacía
3: for iteracion in iteraciones do
4:   for hormiga in hormigas do
5:     start_node ← get_random_start_node()
6:     end_node ← get_random_end_node()
7:     path ← find_path(start_node, end_node)
8:     if path.cost() ≤ mejor_solucion.cost() then
9:       mejor_solucion ← path
10:    end if
11:  end for
12:  Actualización_de_feromonas()
13: end for

```

4.3.1. Inicialización

Durante la fase de inicialización se fija el valor inicial de las casillas de la matriz de feromonas. La inicialización se realiza asignando un valor cercano a cero a cada elemento de la matriz. Esto con el objetivo de garantizar que todos los arcos tengan la misma probabilidad de ser visitados al principio del algoritmo.

4.3.2. Construcción de soluciones

El proceso de construcción de soluciones se lleva a cabo para cada hormiga en cada iteración del algoritmo. En este proceso, cada hormiga encuentra un camino que recorre todos los arcos del grafo, comenzando desde uno de los nodos iniciales y terminando en uno de los nodos finales, pudiendo volver a pasar por un arco previamente recorrido con un costo reducido, según se detalla a continuación.

En la construcción de caminos, cada hormiga elige el siguiente arco a visitar utilizando la regla de transición presentada en la ecuación 18 para determinar la probabilidad de seleccionar cada arco. Los factores clave de la función de transición son τ_{ij} , que representa el valor de las feromonas del arco ij . El factor η_{ij} que corresponde a la heurística relacionada con la visibilidad de las hormigas y se calcula como el inverso del tiempo de viaje del arco ij . Por último, el factor μ_{ij} es un ponderador que indica la reducción en el tiempo de viaje del arco ij una vez esta ya ha sido visitado. Además, cada hormiga tiene un conjunto de arcos ν que corresponden a los arcos permitidos para visitar.

$$P_{ij}^h = \begin{cases} \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta \cdot [\mu_{ij}]^\gamma}{\sum_{k \in \nu} [\tau_{ik}]^\alpha \cdot [\eta_{ik}]^\beta \cdot [\mu_{ik}]^\gamma} & j \in \nu \\ 0 & j \notin \nu \end{cases} \quad (18)$$

Debido a que las hormigas tienen permitido visitar múltiples veces un mismo arco, es necesario incluir un mecanismo que les permita evitar la construcción de ciclos al recorrer los arcos del grafo. Por esta razón, el algoritmo propuesto dota a las hormigas con memoria a corto plazo, lo que permite que las hormigas eviten visitar un arco ya recorrido desde la última vez que realizaron una recolección de desechos. Esta memoria a corto plazo es representada en la ecuación 18 como el conjunto ν que lista los arcos que pueden ser visitados.

En la figura 6, se muestra un ejemplo de cómo una hormiga utiliza su memoria a corto plazo con el objetivo de buscar arcos pendientes por visitar. En el ejemplo, los nodos azules denotan la ubicación actual de la hormiga. Los nodos rojos indican los nodos que la hormiga ya ha visitado y que, por ende, están almacenados en su memoria a corto plazo. Además, se aprecian arcos de color verde y rojo, que representan los arcos visitados y no visitados, respectivamente.

El ejemplo señalado anteriormente, ilustra cómo una hormiga utiliza su memoria a corto plazo para encontrar un arco no visitado. Comienza en el nodo B (Estado inicial) y se mueve hacia el nodo A utilizando las heurísticas mencionadas anteriormente (Movimiento 1). Sin embargo, como el arco BA ya fue visitado, la hormiga agrega el nodo B a su memoria para evitar volver a visitarlo y generar un ciclo. Así, su siguiente movimiento es hacia el único nodo disponible desde su posición actual, es decir, el nodo C (Movimiento 2). Pero nuevamente, como el arco AC también ha sido visitado previamente, la hormiga agrega el nodo A a su memoria a corto plazo. Por lo tanto, solo le queda la opción de dirigirse hacia el nodo D y visitar el arco pendiente CD . Tras haber explorado un arco nuevo, la hormiga borra su memoria a corto plazo, lo que le permite volver a visitar todos los nodos y repetir el proceso nuevamente.

La memoria a corto plazo de las hormigas no solo les permite evitar ciclos, como se señaló anteriormente, sino que también les brinda la capacidad de deshacer un movimiento las veces que sean necesarias cuando no existan arcos que se puedan visitar desde el nodo actual. Durante este proceso, los nodos visitados cuyos movimientos se han deshecho, se mantienen en la memoria a corto plazo para evitar ciclos en futuras elecciones.

Un ejemplo de esto se muestra en la figura 7. Considerando la misma notación que se expuso en el ejemplo anterior, donde los arcos visitados y no visitados se representan en colores verde y rojo respectivamente. El nodo azul representa la ubicación actual de la hormiga y los nodos rojos son los nodos presentes en la memoria de la hormiga. Suponiendo que una hormiga se encuentra en el nodo C (Estado inicial) y realiza un movimiento hacia el nodo A (Movimiento 1). Dado a que el arco CA ya había sido visitado, la hormiga añade el nodo C a su memoria para no volver a visitarlo. Sin embargo, como no existe ningún nodo accesible desde el nodo A , la hormiga deshace su último movimiento, regresando al nodo C pero manteniendo el nodo A en su memoria para no volver a visitarlo. Una vez el movimiento es deshecho, la hormiga vuelve a aplicar las heurísticas para seleccionar un arco a visitar. En este caso, el único posible es el CB (estado final).

Es importante destacar que la memoria de las hormigas se reinicia cada vez que estas realizan una recolección y que los movimientos pueden ser deshechos múltiples veces si es que la hormiga se encuentra en un camino sin nodos por visitar. Estas decisiones se toman con el objetivo de permitir que las hormigas exploren todos los nodos que consideren necesarios para encontrar los arcos no visitados, evitando así la posibilidad de caer en un ciclo.

El algoritmo 2 presenta la estructura general del procedimiento de construcción de soluciones. Comienza con la selección de un nodo para visitar, determinado en la línea 2, utilizando las heurísticas mencionadas previamente y la memoria a corto plazo de la hormiga.

Si no hay un nodo disponible para visitar, el algoritmo deshace el último movimiento, regresando al nodo anterior y finalizando anticipadamente la iteración (líneas 3 y 6).

En caso contrario, el algoritmo verifica si el arco ya ha sido visitado; si es así, añade el nodo a la memoria. En caso contrario, reduce la lista de arcos por visitar y reinicia la memoria de

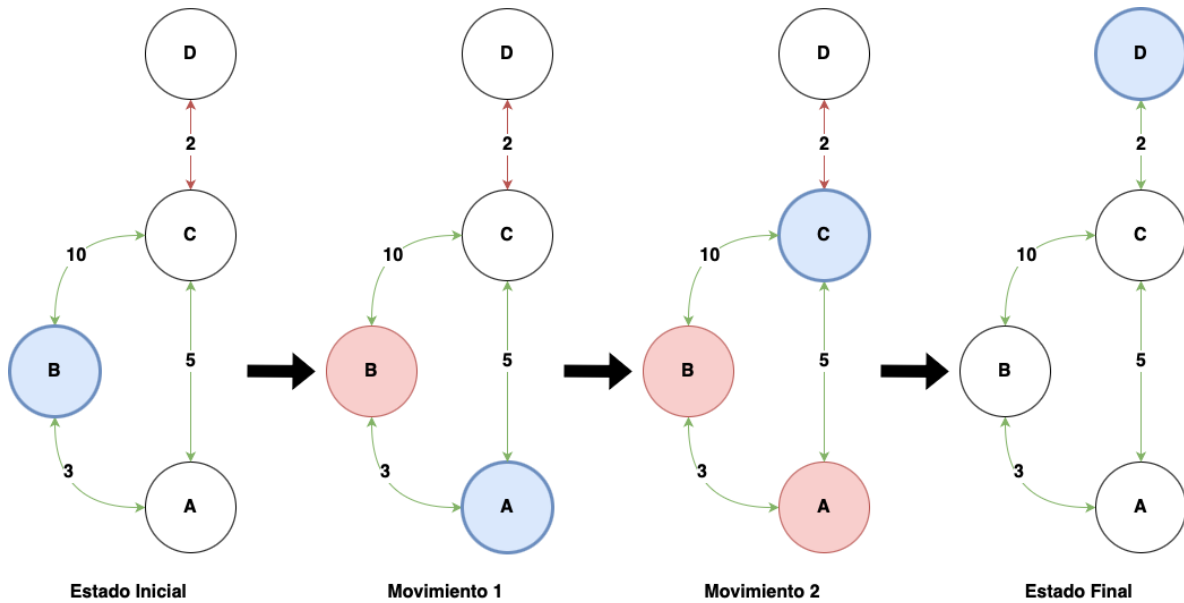


Figura 6: Ejemplo de una hormiga buscando arcos pendientes utilizando su memoria a corto plazo. Fuente: Propia

la hormiga para permitir volver a visitar nodos y/o arcos ya recolectados(líneas 10 y 11).

Finalmente, el algoritmo actualiza el nodo anterior al nodo seleccionado en la iteración actual (línea 13). Todo este procedimiento se repite hasta que todos los arcos hayan sido visitados y la hormiga se encuentre en el nodo final (línea 1).

Algorithm 2 find path

```

1: while arcos_por_visitar > 0 or nodo_anterior ≠ nodo_final do
2:   proximo_nodo ← obtener_proximo_nodo(nodo_anterior, memoria)
3:   if proximo_nodo = nil then
4:     undoLastMovement()
5:     continue
6:   end if
7:   if arco_fue_visitado(nodo_anterior, proximo_nodo) then
8:     agregar_arco_a_memoria(nodo_anterior, proximo_nodo)
9:   else
10:    arcos_por_visitar ← arcos_por_visitar - 1
11:    reiniciar_memoria()
12:   end if
13:   nodo_anterior ← proximo_nodo
14: end while

```

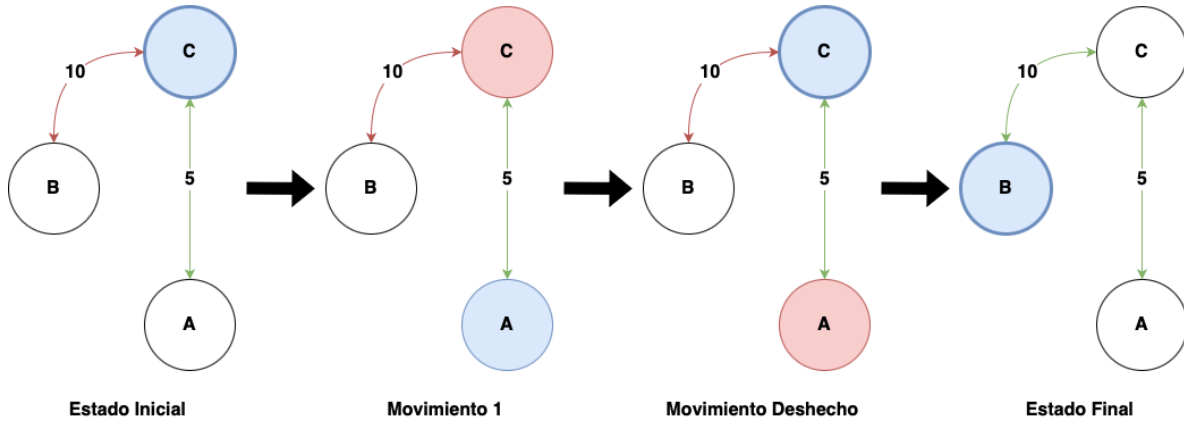


Figura 7: Ejemplo de una hormiga deshaciendo un movimiento utilizando su memoria a corto plazo. Fuente: Propia

4.3.3. Actualización de feromona

El procedimiento de actualización de feromonas del algoritmo se realiza de acuerdo a la ecuación 19. En dicha expresión, ρ representa el factor de evaporación, cuyo valor se encuentra en el rango $0 < \rho \leq 1$. Por su parte, τ_{ij}^{actual} y τ_{ij}^{nuevo} representan, respectivamente, el valor actual y nuevo de las feromonas en el arco ij . Además, se introduce la variable τ_{ij}^h la cual representa el valor de feromonas depositadas en el arco ij por la hormiga h .

$$\tau_{ij}^{nuevo} = (1 - \rho) \cdot \tau_{ij}^{actual} + \sum_{h=0}^M \tau_{ij}^h \quad (19)$$

La cantidad de feromonas que cada hormiga deposita en los arcos que visita esta determinada por la expresión 20. En dicha expresión, Q representa una constante y t_{ij}^k corresponde al tiempo que la hormiga h tardó en recorrer el arco ij en su k -ésimo trayecto. El valor del parámetro Q se estudia en el capítulo de experimentos.

$$\tau_{ij}^h = \sum_{k=0} \frac{Q}{t_{ij}^k} \quad (20)$$

Además, se considera que el nivel de feromonas nunca debe ser menor o igual a 0, por lo tanto, el procedimiento de actualización de feromonas limita los cambios dentro de este rango. Junto con esto, el algoritmo puede reiniciar la matriz de feromonas una vez se completen una cantidad I iteraciones. El valor del parámetro I se estudia en el capítulo de experimentos.

El algoritmo 3 describe el procedimiento de actualización de feromonas. Entre las líneas 1 y 5, se restablecen las feromonas de cada arco a un valor mínimo después de que el algoritmo

ha realizado I iteraciones. Por otro lado, entre las líneas 6 y 16, se lleva a cabo la actualización de feromonas en cada arco, evaporando una parte de las feromonas existentes y luego añadiendo las depositadas por las hormigas durante la iteración actual.

Algorithm 3 Actualización de feromonas

```

1: if  $i \% I == 0$  then
2:   for arco in Arcos do
3:     arco.feromonas = valor_minimo
4:   end for
5: else
6:   feromonas_a_depositar  $\leftarrow$  Solución Vacía
7:   for solucion in Soluciones do
8:     for movimiento in Movimientos do
9:       feromonas_a_depositar[movimiento.arco]  $+= Q/t_{movimiento}$ 
10:    end for
11:  end for
12:  for arco in Arcos do
13:    feromonas  $\leftarrow (1 - \rho) * \text{arco.feromonas} + \text{feromonas\_a\_depositar}[\text{arco}]$ 
14:    arco.feromonas = max(feromonas, valor_minimo)
15:  end for
16: end if

```

4.3.4. Criterios de detención

El algoritmo cuenta con dos criterios de detención. El primero corresponde al número máximo de iteraciones permitidas, mientras que el segundo se relaciona con la cantidad de iteraciones consecutivas en las que el algoritmo no logra encontrar una solución mejor que la actual.

4.3.5. Resumen

Con el fin de abordar la problemática de la recolección de desechos puerta a puerta con ventanas de tiempo, se propone la utilización de un algoritmo basado en la meta-heurística de colonias de hormigas (ACO). El objetivo del algoritmo propuesto es encontrar la ruta más eficiente que atravesase todos los arcos de un grafo, partiendo desde un nodo inicial y culminando en un nodo final, mientras se respeta las ventanas de tiempo establecidas para cada arco. Estas ventanas de tiempo indican los momentos en los cuales es posible transitar por el respectivo arco.

La implementación del algoritmo de colonias de hormigas propuesto se diferencia de los algoritmos ACO tradicionales en que las hormigas son dotadas de memoria a corto plazo para

que puedan volver a recorrer arcos ya visitados anteriormente con el objetivo de encontrar rutas más eficientes. Junto a esto, el algoritmo implementa los parámetros típicos de un algoritmo basado en colonia de hormigas como lo son la visibilidad (η) de las hormigas y las feromonas de los arcos (τ), pero además, el algoritmo propuesto incorpora un parámetro extra llamado μ el cual es un ponderador que reduce los tiempos de viaje de las hormigas en arcos que han sido visitados anteriormente con el objetivo de simular un viaje sin recolección de desechos debido a que ya fueron recolectados anteriormente.

El enfoque de dotar a las hormigas con memoria a corto plazo para volver a recorrer arcos previamente visitados resulta esencial para optimizar aún más la eficiencia del algoritmo propuesto. Esta innovadora capacidad permite a las hormigas aprovechar rutas conocidas y, en conjunto con los parámetros clásicos de visibilidad (η) y feromonas de los arcos (τ), lograr un equilibrio óptimo entre exploración y explotación en el proceso de búsqueda. La adición del parámetro μ , como ponderador para reducir los tiempos de viaje en arcos ya recolectados, agrega una dimensión de realismo al modelo simulando la situación en la que los desechos ya han sido recogidos previamente, optimizando así la eficacia del recorrido y la planificación de las rutas. Con esta combinación de características, el algoritmo de colonias de hormigas adaptado se convierte en una herramienta prometedora para abordar eficazmente el desafío de la recolección de desechos puerta a puerta con ventanas de tiempo.

CAPÍTULO 5

VALIDACIÓN DE LA SOLUCIÓN

El presente capítulo se enfoca en los experimentos y los resultados obtenidos para evaluar el rendimiento del algoritmo propuesto para el problema de enrutamiento de vehículos de recolección de desechos puerta a puerta con ventanas de tiempo. En primer lugar, se detalla el entorno experimental utilizado y se presentan los objetivos del análisis experimental. A continuación, se describe el proceso de ajuste de los parámetros del algoritmo. Posteriormente, se presentan los conjuntos de instancias de prueba empleados en el proceso de evaluación. Finalmente, se exponen los resultados obtenidos y se analizan las conclusiones derivadas de cada uno de los análisis realizados.

5.1. Experimentación

El proceso de experimentación se enfoca en evaluar la eficacia y eficiencia de las rutas de recolección de basura en distintas comunas que es capaz de entregar el algoritmo desarrollado. Estos objetivos se desglosan en tres puntos esenciales:

- **Evaluación de eficacia:** En este punto, se realiza un análisis de la capacidad del algoritmo para identificar rutas de recolección óptimas, minimizando así el tiempo necesario para recoger los desechos. Se busca determinar la habilidad del algoritmo para diseñar rutas que no solo sean rápidas, sino también prácticas y realistas en situaciones reales de recolección de basura.
- **Evaluación de eficiencia:** Este punto se enfoca en medir el rendimiento del algoritmo en términos de velocidad de cálculo y utilización de recursos computacionales. El objetivo es garantizar que las soluciones se generen de manera eficaz y en un tiempo razonable.
- **Análisis de la calidad de las soluciones:** El objetivo de este punto es evaluar detalladamente la calidad de las rutas generadas, considerando criterios como distancia total recorrida y el tiempo empleado en cada ruta.

Para llevar a cabo estas evaluaciones se utilizó una computadora MacBook Pro con 16 GB de memoria RAM y un chip Apple M1 Pro ejecutando el sistema operativo macOS Sonoma. Esta configuración proporcionó un entorno robusto y consistente para las pruebas.

5.2. Sintonización de parámetros

La optimización de los parámetros necesarios para la ejecución del algoritmo se realizó mediante un proceso de ajuste utilizando el *framework* denominado *ParamILS* [Hutter *et al.*, 2009]. Este *framework* genera diversas configuraciones de parámetros a partir de un conjunto predefinido de valores, los cuales se detallan en la tabla 7. El procedimiento se inicia con una configuración inicial y, posteriormente, se procede a ajustar los parámetros de manera individual, conservando las configuraciones que mejoran los resultados en términos de calidad obtenida en los experimentos.

Parámetro	Valores Posibles	Valores Discretizados
Hormigas	$(0, \infty)$	0, 5, 10, 20, 30, 50, 100
Iteraciones	$(0, \infty)$	50, 100, 200, 500, 1000
Máximas iteraciones sin mejora	$(0, \infty)$	5, 10, 20, 50
α	$[0.0, \infty)$	1, 2, 5, 10
β	$[0.0, \infty)$	1, 2, 5, 10
γ	$[0.0, \infty)$	1, 2, 5, 10
Coeficiente de evaporación	$[0.0, 1.0]$	0, 0.5, 1.0
Coeficiente de reducción	$[0.0, 1.0)$	0, 0.25, 0.50, 0.75, 0.95
Factor Q	-	0.5, 1.0
I	$[0, \text{Iteraciones}]$	-

Tabla 7: Parámetros y sus valores posibles

5.2.1. Resultados Sintonización

Después de realizar la sintonización del algoritmo utilizando los conjuntos de valores para cada parámetro, se han identificado dos configuraciones que obtuvieron los mejores resultados. Estas configuraciones, designadas como configuración 1 y configuración 2, se detallan en la tabla 8.

Las diferencias entre ambas configuraciones se centran en dos parámetros clave: coeficiente de Evaporación e iteraciones. En la configuración 1, el coeficiente de evaporación está establecido en 0.5. Este valor indica que, en cada iteración, el algoritmo eliminará la mitad de las feromonas previamente depositadas. Por el contrario, para la configuración 2, este parámetro se fija en 1.0, lo que implica que el algoritmo elimina por completo todas las feromonas entre iteraciones. Esta situación conlleva a que las hormigas trabajen para obtener una solución sin contar con información alguna de las iteraciones previas.

Parámetro	Configuración 1	Configuración 2
Hormigas	100	100
α	1	1
β	1	1
Coefficiente de Evaporación	0.5	1
γ	5	5
Iteraciones	500	1000
Factor Q	0.5	0.5
I	500	1000
Coefficiente de reducción de arcos visitados	0.95	0.95
Máximas iteraciones sin mejora	50	50

Tabla 8: Tabla de comparación entre las dos configuraciones seleccionadas.

5.3. Instancias de prueba

En esta sección se presenta el formato de las instancias de pruebas utilizado por el algoritmo y el proceso de generación de casos de pruebas basado en información real de las comunas de Renca, Quinta Normal y Cerro Navia de Santiago de Chile.

5.3.1. Formato

Cada instancia de prueba es representada por una cantidad de nodos, nodos iniciales, nodos finales, una matriz de tiempos de viaje de cada arco y otras dos matrices que representan las ventanas de tiempo de cada arco. Un ejemplo del formato de casos de pruebas se presenta en la figura 8.

```

Dimension
Dimension nodos iniciales
Nodos iniciales
Dimension nodos finales
Nodos finales
Matriz de tiempos de viaje
Matriz de tiempos de inicio
Matriz de tiempos de termino

```

Figura 8: Formato instancias de pruebas

A continuación se muestra un ejemplo de instancia de prueba. La figura 9 muestra el archivo de datos de la instancia 6. Esta instancia tiene cuatro nodos que se indican como dimensión. La instancia cuenta con un nodo inicial y un nodo final que corresponden a los nodos A (0) y D (3) respectivamente. Junto a lo anterior, en el archivo de datos de la instancia se presenta

una matriz de tiempos de viaje con los valores indicados en la figura 6. En esta matriz los arcos marcados con un valor de -1 indican que el arco es no dirigido y se puede recolectar los desechos de dicha calle en ambos sentidos. Por último, se presentan dos matrices que indican el inicio y término de las ventanas de tiempo de cada arco.

```

4          # Dimension
1          # Dimension nodos iniciales
0          # Nodos iniciales (En este caso sería el nodo A)
1          # Dimension nodos finales
3          # Nodos finales (En este caso sería el nodo D)
0 3 5 0    # Tiempos de viaje del nodo 0 al nodo 0, 1, 2 y 3
-1 0 10 0  # Tiempos de viaje del nodo 1 al nodo 0, 1, 2 y 3
-1 -1 0 2   # Tiempos de viaje del nodo 2 al nodo 0, 1, 2 y 3
0 0 -1 0    # Tiempos de viaje del nodo 2 al nodo 0, 1, 2 y 3
0 0 0 0     # Inicio ventana del nodo 0 al nodo 0, 1, 2 y 3
0 0 0 0     # Inicio ventana del nodo 1 al nodo 0, 1, 2 y 3
0 0 0 0     # Inicio ventana del nodo 2 al nodo 0, 1, 2 y 3
0 0 0 0     # Inicio ventana del nodo 3 al nodo 0, 1, 2 y 3
50 50 50 50 # Termino ventana del nodo 0 al nodo 0, 1, 2 y 3
50 50 50 50 # Termino ventana del nodo 1 al nodo 0, 1, 2 y 3
50 50 50 50 # Termino ventana del nodo 2 al nodo 0, 1, 2 y 3
50 50 50 50 # Termino ventana del nodo 3 al nodo 0, 1, 2 y 3

```

Figura 9: Ejemplo de instancia de prueba de cuatro nodos

5.3.2. Generación de casos

Para la generación de casos se utiliza la información de las redes viales de las regiones de Chile disponibles en *OpenStreetMap*. En este trabajo de título se implementó un algoritmo que utiliza el formato *GeoJSON* [Butler *et al.*, 2016] para generar los casos de prueba. Este formato es un estándar abierto diseñado para representar características geográficas utilizando distintas geometrías, como puntos, líneas y polígonos y se basa en el formato *JSON*.

La utilización del formato *GeoJSON* para la generación de casos pruebas permite tener un control mayor en la selección de áreas geográficas en donde se desea probar el algoritmo.

En el marco de este trabajo, se presentan tres conjuntos de casos de prueba, cada uno asociado a una comuna de interés para el estudio. Las comunas seleccionadas para la investigación son las comunas de la Región Metropolitana: Renca, Cerro Navia y Quinta Normal. Estas comunas han sido elegidas debido a su idoneidad para el análisis de procesos de recolección de residuos colaborativos. Esto se debe a que las tres comunas comparten límites geográficos.

Para cada una de estas comunas, se han diseñado tres instancias de pruebas diferentes: una

de tamaño pequeño, otra de tamaño mediano y una tercera de tamaño grande. Las instancias de tamaño pequeño abarcan entre 31 y 59 nodos y entre 73 y 147 arcos. Las instancias de tamaño mediano abarcan entre 117 y 173 nodos y entre 338 y 439 arcos. Por último, las instancias de tamaño grande tienen entre 181 y 285 nodos y entre 525 y 723 arcos.

5.3.3. Caso 1: Renca

El caso 1 corresponde a zonas de la comuna de Renca. La comuna de Renca destaca por su diversidad geográfica y demográfica, lo que lo convierte en un desafío interesante para evaluar la efectividad del algoritmo desarrollado.

La primera instancia de prueba se puede observar en la figura 10 y tiene un tamaño aproximado de 5 hectáreas. La instancia considera un total de 31 nodos (intersecciones) y 73 arcos (calles) que requieren recolección de residuos. Esta sección está limitada por la calle Volcán Aconcagua por arriba, la Avenida José Miguel Infante por abajo y por los lados el pasaje Nuevo y Avenida Condell.

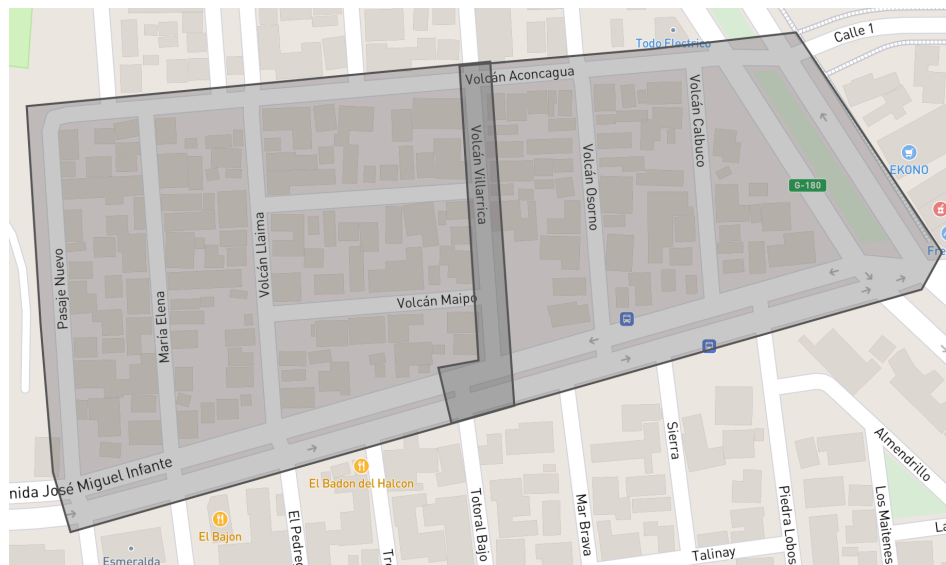


Figura 10: Sección 1 (tamaño pequeño)

La segunda instancia de pruebas corresponde a la presentada en la figura 11. La instancia tiene un tamaño aproximado de 24 hectáreas. Contiene 117 intersecciones y 338 calles y está limitada por arriba y los lados por las mismas calles que la instancia anterior y por abajo por la calle Topocalma. Adicionalmente, por la izquierda también está limitada por la calle Coloso.

La última instancia de pruebas se presenta en la figura 12. Esta instancia posee un tamaño de 47 hectáreas aproximadamente. Este caso posee 181 intersecciones y 525 calles y está

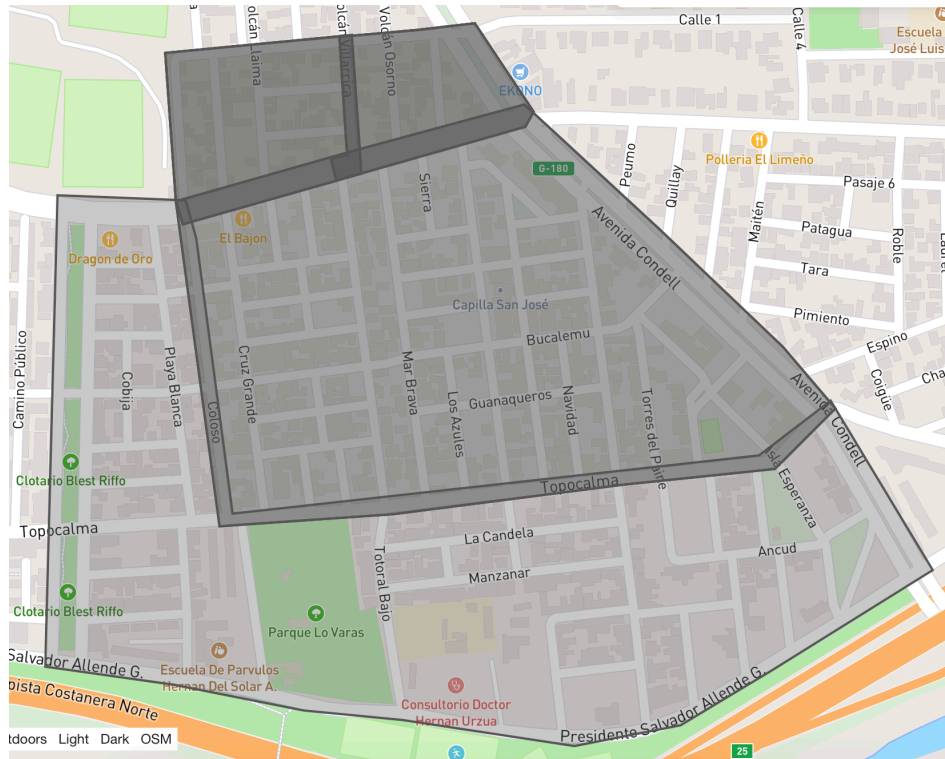


Figura 12: Sección 3 (tamaño grande)

táreas limitada por el norte, sur y este por las mismas calles que la instancia anterior y por la calle Santos Luis Medel por el oeste. El número de intersecciones de esta instancia es de 214 con 644 calles.

5.3.5. Caso 3: Quinta Normal

La última comuna con la que se experimentará corresponde a la de Quinta Normal. Esta comuna es limítrofe a las dos comunas presentadas anteriormente.

La instancia de tamaño pequeño de este caso se presenta en la figura 16. El caso tiene 59 intersecciones con 147 calles y se encuentra limitado por la Avenida Carrascal por abajo, por la derecha por la calle Brisas del Río y por arriba y la izquierda por la Avenida Costanera Sur. La superficie aproximada de esta instancia es de 19 hectáreas.

La instancia mediana se presenta en la figura 17. La instancia tiene una superficie aproximadamente de 61 hectáreas y cuenta con 173 intersecciones y 493 calles. Esta zona esta limitada por arriba por la Costanera Sur, por la izquierda por la instancia anterior, por abajo por la Avenida Carrascal y por la derecha por Samuel Izquierdo. En esta instancia, se observa que se omite un área localizada cerca de la Escuela España. Esta exclusión se debe a que, aunque el algoritmo es compatible con rotondas y varios tipos de intersecciones, no admite



Figura 13: Sección 3 (tamaño pequeño)

un tipo específico de calle que se conecta consigo misma, como es el caso de esta vía. Esta limitación conduce a la exclusión de esta zona del análisis.

La instancia más grande tiene 285 intersecciones con 723 calles y se observa en la figura 18. Esta instancia cuenta con una superficie de 113 hectáreas y se encuentra limitada por la izquierda por la instancia mediana, por abajo por la Avenida Carrascal y por la derecha y arriba por la calle Lo Espinoza.

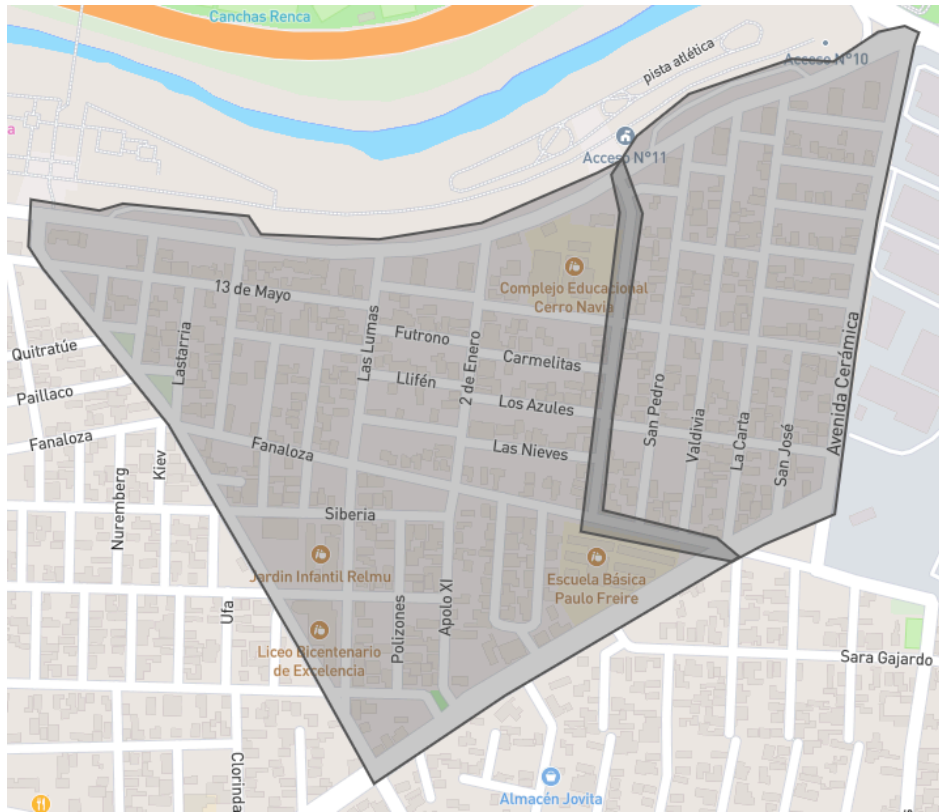


Figura 14: Sección 3 (tamaño mediano)

5.4. Resultados

En esta sección se exponen y analizan los resultados obtenidos utilizando el algoritmo propuesto. Se presentarán detalladamente los hallazgos significativos que surgieron durante la obtención de los resultados, abordando cada uno de los objetivos planteados y respondiendo a las preguntas de investigación formuladas anteriormente. Estos resultados se estructuran en función de las comunas de donde fueron obtenidos.

Los resultados presentados en esta sección fueron obtenidos tras ejecutar el algoritmo propuesto 10 veces para cada caso de pruebas con cada configuración de parámetros, lo que generó un total de 540 soluciones finales. Para cada caso, se muestran tres tablas cuya estructura se detallada a continuación:

1. Tabla de soluciones
2. Tabla de estadísticas
3. Tabla de tiempos

La primera tabla resumen presenta la mejor solución (*Min Solución*) y el promedio de calidad



Figura 15: Sección 3 (tamaño grande)

soluciones (*Prom. Soluciones*) obtenidas por el algoritmo. Además, en caso de que la mejor solución sea infactible, es decir, que la solución sea mayor a la ventana de tiempo, esta estará señalada con un color rojo.

La segunda tabla indica el promedio de la iteraciones en la que el algoritmo encontró la mejor solución (*Prom. Iteraciones*) y el promedio de uso de las calles (*Prom. Uso*) la cual corresponde a la proporción de la cantidad de veces en la que el algoritmo vuelve a pasar por una calle ya recolectada (Un valor de 1.5 señala que el algoritmo recorrió un 50 % de calles extra).

La tercera y última tabla resumen, señala los tiempos de ejecución del algoritmo. Esta tabla incluye el tiempo de ejecución para encontrar la mejor solución (*T.E. Min Solución*) y el promedio de tiempos de ejecución junto a su desviación estándar (*Prom. T.E.*).

Es importante señalar que la columna que indica la proporción de uso de calles será 1 solo en una instancia ideal, en donde se pueda recorrer todo el mapa sin volver a pasar por calles ya visitadas como lo que se da en problemas como el *Traveling salesman problem*.



Figura 16: Quinta Normal: Sección 1 (tamaño pequeño)

5.4.1. Caso 1: Renca

Los resultados obtenidos por el algoritmo propuesto para las instancias de la comuna de Renca muestran que pudo encontrar soluciones factibles para los tres tamaños disponibles (pequeño, mediano, grande) en al menos un nivel de ventanas de tiempo, como se muestra en la tabla 9.

En cuanto a las mejores soluciones encontradas por el algoritmo, se observa que son muy similares entre ambas configuraciones. No obstante, la configuración 1 logró mejores resultados en 4 de las 6 soluciones factibles, mientras que la configuración 2 solo obtuvo un resultado mejor en el caso con un tamaño mediano y una ventana de tiempo de 28800[s]

La diferencia principal que explica los resultados entre las dos configuraciones seleccionadas radica en el coeficiente de evaporación. En la configuración 2, este coeficiente tiene un valor de 1, lo que implica que todas las feromonas se evaporan entre iteraciones. Esta condición hace que las hormigas funcionen sin la información de las iteraciones pasadas, por lo que los resultados de previas iteraciones no influirán en las siguientes iteraciones, lo cual dificulta la obtención de soluciones óptimas. A pesar de que el caso en general era relativamente sencillo, las diferencias entre las dos configuraciones no fueron tan notables. Sin embargo, la configuración 1 demostró un mejor rendimiento en general, ya que permitía a las hormigas transmitir información entre iteraciones mediante el uso de sus feromonas.

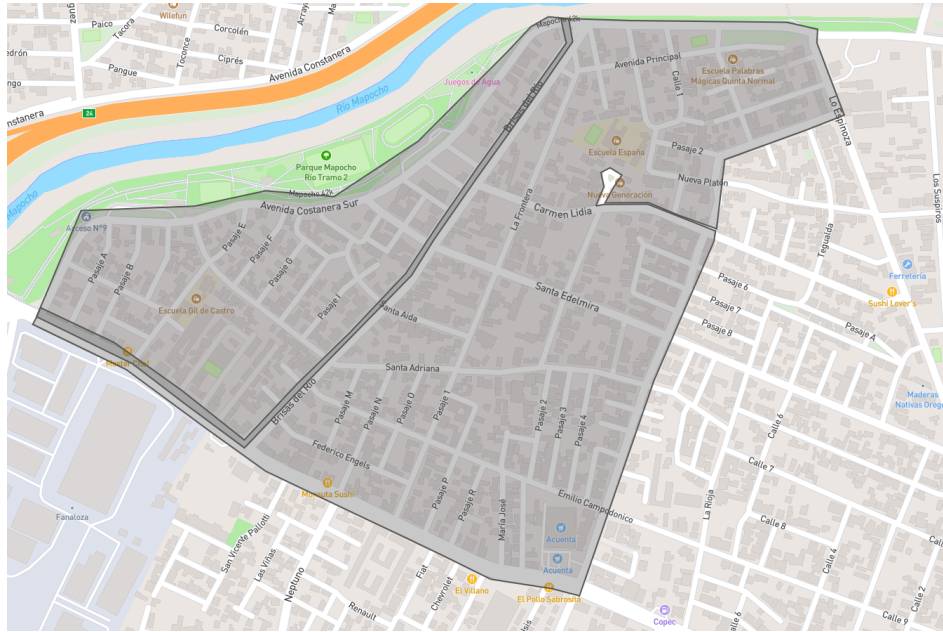


Figura 17: Quinta Normal: Sección 2 (tamaño mediano)

Continuando con el análisis, el promedio de soluciones encontradas por el algoritmo para las instancias es bastante similar entre ambas configuraciones de parámetros. No obstante, al igual que en el caso de la mejor solución, la configuración 1 obtuvo un mejor promedio en 4 de las 6 soluciones factibles. Este resultado se debe a la misma razón que señalo en el párrafo anterior, las feromonas influyen en la generación de los resultados.

En relación a los casos en los que el algoritmo no pudo encontrar una solución factible, como en la instancia mediana con una ventana de 7200s (2 horas), la instancia grande con una ventana de 7200s (2 horas) y la instancia grande con una ventana de 21600s (6 horas). Se observa que el algoritmo presentó un comportamiento llamativo, generando soluciones muy diversas entre sí, lo que explica la alta desviación estándar para estos casos. Este comportamiento se atribuye a las penalizaciones impuestas por el incumplimiento de las ventanas de tiempo, lo cual incide directamente en el aumento de la duración de las soluciones generadas por el algoritmo. La presencia de estas penalizaciones no solo influye en la longitud de las soluciones, sino que también contribuye a una mayor diversificación en los resultados obtenidos, generando una gama más amplia de posibles soluciones debido a la búsqueda de alternativas que eviten o minimicen dichas penalizaciones.

Con respecto a la tabla 10, se puede observar que el promedio de iteraciones en las que el algoritmo encuentra la mejor solución difiere significativamente entre las dos configuraciones. En la primera, los valores tienden a ser más altos en comparación con la configuración 2, llegando a casi duplicar el número de iteraciones necesarias para encontrar la mejor solución para algunos casos. La explicación de este comportamiento se encuentra relacionado al coeficiente de evaporación y se debe a que la configuración 1, a diferencia de la configuración 2, no elimina por completo todas las feromonas entre iteraciones. Esta diferencia en

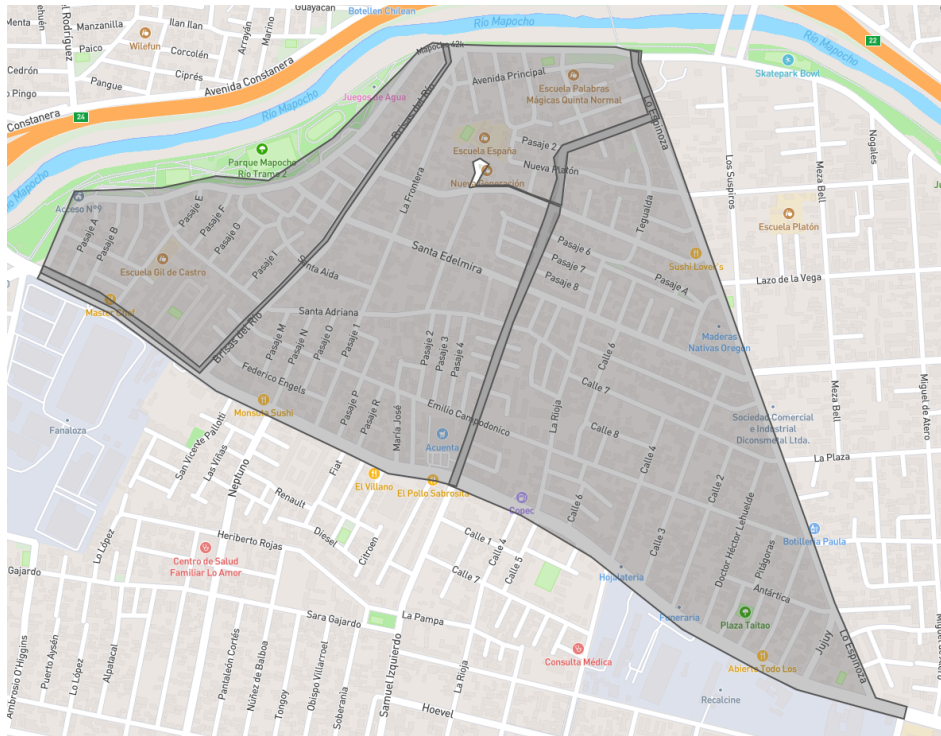


Figura 18: Sección 3 (tamaño grande)

el manejo de las feromonas conlleva un mayor esfuerzo (*overhead*) al utilizar la configuración 1, ya que esta configuración hará que el algoritmo tenga que tomar en consideración la información de las soluciones anteriores para generar las nuevas soluciones.

En relación al uso de calles, según las soluciones obtenidas por el algoritmo y presentadas en la tabla 10, se observa que la configuración 1 tiende a tener valores más bajos en comparación con la configuración 2. Esta última superó a la primera en solo tres de los nueve casos analizados. En cuanto a la desviación estándar del uso de calles, se destaca que ambas configuraciones mantienen valores bastante bajos y muy similares. Sin embargo, en este caso, la configuración 2 tiende a presentar una desviación estándar más baja que la configuración 1.

El uso de las calles se ve principalmente afectado por el Coeficiente de Evaporación en este caso particular. La configuración 2, al evaporar todas las feromonas depositadas por las hormigas entre iteraciones, incita a una mayor repetición de rutas, lo que conduce a un mayor nivel de utilización en comparación con la configuración 1. En contraste, en la configuración 1, las feromonas permiten a las hormigas optimizar hasta cierto punto sus rutas, lo que reduce la tendencia a repetir caminos.

Acerca de los tiempos de ejecución del algoritmo, se puede apreciar en la tabla 11 que la configuración 1 logró obtener su mejor solución en un menor tiempo de ejecución en cuatro de las nueve variaciones del caso, mientras que la configuración 2 lo hizo en cinco de las

variaciones. Además, es importante señalar que, aparentemente, el comportamiento del algoritmo para casos infactibles no incrementa el tiempo de ejecución del mismo, ya que entre las variaciones factibles o infactibles no se aprecian diferencias significativas en términos de tiempo de ejecución.

En cuanto al promedio de tiempos de ejecución del algoritmo para ambas configuraciones, se puede señalar que existe una leve diferencia, aunque siempre manteniendo sus valores en el mismo orden de magnitud. No obstante, se aprecia que la configuración 2 tiende a tener un promedio de tiempos de ejecución más bajo en comparación con la configuración 1. Esto probablemente se pueda explicar debido a que, como señalamos anteriormente, la configuración 1 tuvo un promedio de iteraciones más alto para encontrar sus mejores soluciones que la configuración 2.

Tamaño	Ventana[s]	Configuración1		Configuración2	
		Min Solución[s]	Prom. Soluciones[s]	Min Solución[s]	Prom. Soluciones[s]
Pequeño	7200	3425,6	3450,5 $\pm$ 10,0	3425,6	3447,5 $\pm$ 7,8
	21600	3444,6	3451,4 $\pm$ 4,6	3449,5	3453,9 $\pm$ 5,0
	28800	3425,6	3449,2 $\pm$ 9,7	3448,1	3452,9 $\pm$ 6,1
Mediano	7200	1432270,8	1604222,4 $\pm$ 82448,5	1503546,1	1626441,4 $\pm$ 61682,7
	21600	16626,6	16673,7 $\pm$ 32,8	16638,4	16677,9 $\pm$ 16,1
	28800	16620,6	16680,4 $\pm$ 25,6	16607,6	16677,4 $\pm$ 29,1
Grande	7200	7003487,0	7572961,6 $\pm$ 324304,6	6749401,3	7387157,4 $\pm$ 354411,9
	21600	958635,3	1042900,3 $\pm$ 76937,0	740481,8	919456,1 $\pm$ 96864,7
	28800	28410,1	28516,8 $\pm$ 56,8	28432,0	28541,4 $\pm$ 47,0

Tabla 9: Resumen soluciones: Comuna de Renca

Tamaño	Ventana[s]	Configuración1		Configuración2	
		Prom. Iteraciones[s]	Prom. Uso	Prom. Iteraciones[s]	Prom. Uso
Pequeño	7200	31,4 $\pm$ 24,9	1,21 $\pm$ 0,07	13,2 $\pm$ 16,1	1,19 $\pm$ 0,05
	21600	19,5 $\pm$ 32,2	1,23 $\pm$ 0,05	16,4 $\pm$ 22,6	1,25 $\pm$ 0,05
	28800	11,6 $\pm$ 16,6	1,20 $\pm$ 0,09	25,5 $\pm$ 31,5	1,23 $\pm$ 0,05
Mediano	7200	37,4 $\pm$ 16,9	1,55 $\pm$ 0,06	35,1 $\pm$ 24,8	1,53 $\pm$ 0,08
	21600	52,3 $\pm$ 46,2	1,48 $\pm$ 0,04	30,6 $\pm$ 24,6	1,51 $\pm$ 0,02
	28800	60,3 $\pm$ 40,0	1,51 $\pm$ 0,03	37,6 $\pm$ 26,9	1,51 $\pm$ 0,05
Grande	7200	38,8 $\pm$ 29,1	1,59 $\pm$ 0,06	34,6 $\pm$ 32,7	1,58 $\pm$ 0,05
	21600	34,3 $\pm$ 29,1	1,68 $\pm$ 0,14	44,3 $\pm$ 23,5	1,71 $\pm$ 0,12
	28800	66,9 $\pm$ 34,2	1,53 $\pm$ 0,06	31,1 $\pm$ 24,5	1,54 $\pm$ 0,04

Tabla 10: Resumen estadísticas: Comuna de Renca

Tamaño	Ventana[s]	Configuración1		Configuración2	
		T.E.Min Solución[s]	Prom.T.E.[s]	T.E.Min Solución[s]	Prom.T.E.[s]
Pequeño	7200	35	27, 1 ± 8, 3	16	20, 4 ± 5, 4
	21600	17	23, 1 ± 10, 3	26	21, 2 ± 7, 2
	28800	20	20, 6 ± 6, 3	16	24, 1 ± 10, 1
Mediano	7200	725	726, 5 ± 139, 4	996	801, 2 ± 218, 2
	21600	1722	854, 0 ± 387, 5	643	769, 0 ± 224, 2
	28800	647	920, 4 ± 335, 4	752	842, 0 ± 358, 0
Grande	7200	2197	1880, 3 ± 703, 7	1084	1793, 3 ± 639, 6
	21600	1110	1781, 6 ± 678, 4	1976	2026, 8 ± 581, 1
	28800	2550	2439, 2 ± 746, 1	1821	1738, 5 ± 515, 6

Tabla 11: Resumen tiempos de ejecución: Comuna de Renca

5.4.2. Caso 2: Cerro Navia

Las instancias para la comuna de Cerro Navia se caracterizaban por ser un poco más grandes que las de la comuna de Renca, con un tamaño aproximadamente entre un 5 % y un 97 % mayor dependiendo de la instancia. Esta diferencia, sin duda, afectó los resultados obtenidos por el algoritmo. Como se puede observar en la tabla 12, el algoritmo no pudo encontrar una solución factible en cinco de los nueve casos. Además, para el tamaño grande, el algoritmo no logró encontrar una solución en ninguna ventana de tiempo.

En los casos en que el algoritmo pudo encontrar una solución factible, se observa que, al igual que para la comuna de Renca, los resultados no presentaban diferencias significativas entre las dos configuraciones. Sin embargo, para tres de los cuatro casos, la configuración 1 tuvo un desempeño ligeramente mejor en comparación con la configuración 2, aunque esta diferencia es mínima.

En los casos en los que el algoritmo no encontró soluciones factibles, se observan más diferencias entre las configuraciones. Esto se explica principalmente por las penalizaciones, ya que estas incrementan por cada atraso o adelantamiento con respecto a las ventanas de tiempo de los arcos, inflando el resultado final. En relación a la diferencia entre las dos configuraciones para las soluciones infactibles, se tiene que la configuración 1 obtuvo mejores soluciones para las instancias de tamaño pequeño con una ventana de tiempo de 7200[s] y tamaño grande con una ventana de tiempo de 7200[s]. Mientras que la configuración 2 obtuvo un mejor desempeño en las instancias de tamaño mediano con una ventana de 7200[s] y los casos de tamaño grande con ventanas de tiempo de 21600[s] y 28800[s].

En cuanto al promedio de las soluciones obtenidas por el algoritmo para los casos factibles, se puede observar que para el tamaño pequeño, la configuración 1 tuvo un mejor promedio de soluciones, mientras que la configuración 2 lo tuvo para el tamaño mediano. En relación a las desviaciones estándar de los promedios de las soluciones, ocurre lo contrario. La configuración 1 tuvo una mayor desviación estándar para el tamaño pequeño, mientras que la configuración 2 tiene una mayor desviación estándar para el tamaño mediano.

Al analizar las estadísticas del algoritmo las cuales se presentan en la tabla 13 se puede observar que para la configuración 1, el promedio de iteraciones necesarias para obtener las soluciones es casi el doble entre el tamaño pequeño y mediano, mientras que para la configuración 2 el promedio se mantiene relativamente estable en un rango de valores entre 50 a 65 iteraciones aproximadamente, sin importar el tamaño. Esto último se explica por el coeficiente de evaporación ya que al evaporar todas las feromonas entre iteraciones el algoritmo no puede construir soluciones mas complejas utilizando los cálculos realizados por las hormigas en iteraciones anteriores, por lo que llega siempre a un límite en el que no puede encontrar una solución mejor.

Si observamos el promedio de uso obtenido por las dos configuraciones presentadas en la tabla 13, podemos notar que, una vez más, ambas configuraciones obtienen resultados muy similares entre sí, variando entre uno a tres puntos porcentuales en el promedio de uso de las calles. Sin embargo, la configuración 1 muestra valores más bajos en comparación a la configuración 2 para los casos medianos, pero esta tendencia se invierte para los casos pequeños. Respecto a la desviación estándar de esta medida, se observa una variación mínima entre ambas configuraciones, pero nuevamente se repite la tendencia mencionada anteriormente, la configuración 1 muestra mejores resultados para los casos medianos en comparación con la configuración 2, mientras que esto se invierte para los casos pequeños.

En relación a los tiempos de ejecución presentados en la tabla 14, se observa que el tiempo necesario para obtener la mejor solución en el caso de tamaño pequeño oscila entre 100 y 200 segundos aproximadamente, en donde la configuración 1 logró las mejores soluciones en un tiempo menor en comparación con la configuración 2. Para los casos de tamaño mediano, se aprecia un comportamiento diferente, la configuración 1 muestra mejores resultados para la ventana de tiempo de 21600 segundos, mientras que la configuración 2 destaca para la de 28800 segundos. En cuanto al promedio, se evidencia una clara tendencia, la configuración 1 ofrece mejores resultados para el caso pequeño, mientras que la configuración 2 lo hace para el caso mediano. Además, al analizar la desviación estándar de la configuración 2 se observa que esta tiende a ser más alta en relación al promedio para los cuatro casos, mientras que la configuración 1, aunque varía, mantiene una mayor estabilidad.

La diferencia mencionada anteriormente en relación al promedio de los tiempos de ejecución entre ambas configuraciones, se debe al coeficiente de evaporación. Ya que las feromonas estén generando un costo adicional (*overhead*) en el proceso de búsqueda de soluciones. Por lo tanto, en los casos más grandes, el algoritmo 1 requiere realizar más cálculos, lo que resulta en tiempos de ejecución más prolongados en comparación con la configuración 2, que no utiliza feromonas para calcular las soluciones.

Tamaño	Ventana[s]	Configuración1		Configuración2	
		Min Solución[s]	Prom. Soluciones[s]	Min Solución[s]	Prom. Soluciones[s]
Pequeño	7200	17442,8	18236, 8 ± 707, 8	17667,6	18689, 3 ± 682, 0
	21600	8158,7	8174, 5 ± 9, 0	8164,1	8173, 5 ± 6, 1
	28800	8154,7	8175, 4 ± 14, 1	8160,0	8172, 1 ± 7, 5
Mediano	7200	2252326,0	2463648, 7 ± 99911, 7	2240647,5	2390627, 3 ± 127641, 9
	21600	21317,6	21349, 8 ± 17, 3	21334,3	21366, 6 ± 20, 3
	28800	21339,5	21364, 9 ± 16, 1	21324,5	21369, 5 ± 23, 9
Grande	7200	15378439,3	16818333, 4 ± 1086317, 5	15488255,7	16722668, 7 ± 784606, 3
	21600	4804179,5	5377426, 4 ± 422945, 5	4387821,1	5211703, 8 ± 396910, 8
	28800	1381241,5	1819165, 7 ± 273249, 8	1526858,0	1854811, 4 ± 234893, 5

Tabla 12: Resumen soluciones: Comuna de Cerro Navia

Tamaño	Ventana[s]	Configuración1		Configuración2	
		Prom. Iteraciones	Prom. Uso	Prom. Iteraciones	Prom. Uso
Pequeño	7200	50, 1 ± 26, 4	1, 34 ± 0, 06	43, 8 ± 23, 3	1, 35 ± 0, 06
	21600	38, 9 ± 19, 1	1, 29 ± 0, 03	62, 1 ± 28, 0	1, 28 ± 0, 02
	28800	45, 0 ± 29, 2	1, 29 ± 0, 05	48, 4 ± 40, 4	1, 27 ± 0, 02
Mediano	7200	52, 4 ± 23, 4	1, 46 ± 0, 05	46, 8 ± 25, 7	1, 41 ± 0, 04
	21600	83, 3 ± 34, 3	1, 39 ± 0, 02	66, 2 ± 34, 6	1, 42 ± 0, 04
	28800	80, 5 ± 23, 6	1, 42 ± 0, 03	53, 6 ± 23, 8	1, 43 ± 0, 04
Grande	7200	33, 9 ± 26, 3	1, 75 ± 0, 08	40, 6 ± 28, 2	1, 76 ± 0, 06
	21600	42, 9 ± 29, 2	1, 84 ± 0, 12	54, 2 ± 23, 7	1, 80 ± 0, 07
	28800	40, 1 ± 24, 2	1, 89 ± 0, 08	36, 0 ± 26, 6	1, 84 ± 0, 06

Tabla 13: Resumen estadísticas: Comuna de Cerro Navia

Tamaño	Ventana[s]	Configuración1		Configuración2	
		T.E.Min Solución[s]	Prom.T.E.[s]	T.E.Min Solución[s]	Prom.T.E.[s]
Pequeño	7200	137	137, 7 ± 35, 0	180	127, 2 ± 31, 2
	21600	109	122, 3 ± 24, 2	182	151, 2 ± 37, 2
	28800	191	130, 9 ± 38, 1	208	133, 0 ± 54, 0
Mediano	7200	1162	954, 4 ± 215, 1	1273	1001, 2 ± 282, 9
	21600	1004	1229, 5 ± 308, 7	1030	1179, 4 ± 358, 8
	28800	1166	1203, 7 ± 220, 0	897	1064, 4 ± 244, 5
Grande	7200	3980	3355, 5 ± 1005, 5	3706	3589, 8 ± 1014, 3
	21600	4113	3691, 5 ± 1136, 8	3096	4093, 3 ± 847, 4
	28800	3517	3601, 1 ± 907, 6	5019	3417, 0 ± 1071, 0

Tabla 14: Resumen tiempos de ejecución: Comuna de Cerro Navia

5.4.3. Caso 3: Quinta Normal

El caso de Quinta Normal es mas llamativo en relación a los anteriores, esto se debe a que el algoritmo solo pudo encontrar soluciones factibles para el caso de tamaño pequeño de esta comuna y, además, solo utilizando las ventanas de tiempo de 21600 y 28800 segundos.

Estos resultados se explican principalmente por la diferencia en los tamaños de los casos de esta comuna en comparación a los de las anteriores, ya que los casos en general eran entre un 38 % y hasta un 100 % más grandes con respecto a la comuna de Renca, por lo que era esperable que el algoritmo no pudiera encontrar soluciones factibles para ciertos casos. Sin embargo, esto no deja de ser interesante para analizar el comportamiento del algoritmo.

Si se analizan las cuatro soluciones que encontró el algoritmo, presentadas en la tabla 15, no se encuentran diferencias significativas. Aun así, para el caso de la ventana de tiempo de 21600 segundos, la configuración 2 tuvo un mejor resultado mientras que la configuración 1 lo tuvo en la ventana de tiempo de 28800 segundos. Sin embargo, como se señaló anteriormente, esta diferencia mínima.

El promedio de soluciones obtenidas por el algoritmo, se puede notar que para el caso con una ventana de tiempo de 21600 segundos es casi idéntico entre ambas configuraciones. Sin embargo, la desviación estándar cambia levemente, siendo un poco mayor en la configuración 2. Para el caso de la ventana de tiempo de 28800 segundos la configuración 1 obtiene mejores resultados pero con una desviación estándar mas alta en comparación a la configuración 2.

Al analizar el promedio de iteraciones indicado en la tabla 16, se evidencia que la configuración 2 generó soluciones para las ventanas de tiempo de 21600 y 28800 segundos en alrededor de 25 y 22 iteraciones, respectivamente, con una desviación estándar entre 17 y 35, respectivamente. Estos valores son inferiores en comparación con la configuración 1, que solo supera a la configuración 2 en el promedio de iteraciones para la ventana de tiempo de 28800 segundos. Respecto al promedio de utilización de las calles, también reflejado en la misma tabla, se observa que la configuración 2 tiende a exhibir desviaciones estándar más reducidas en contraste con la otra configuración. Sin embargo, esta última configuración logró soluciones con un promedio de uso más bajo (1,33) para la ventana de tiempo de 28800 segundos.

La razón por la cual la Configuración 2 generalmente presenta desviaciones estándar más bajas en comparación con la Configuración 1 se debe a que la segunda configuración evapora todas las feromonas entre iteraciones. Como resultado, las soluciones generadas dependen en mayor medida de otras heurísticas, las cuales tienden a producir resultados más similares.

Al considerar los tiempos de ejecución obtenidos por el algoritmo al utilizar ambas configuraciones, los cuales son detallados en la tabla 17. Se observa que ambas configuraciones alcanzan sus mejores resultados respectivos en el mismo tiempo (117 segundos). Con respecto al promedio de los tiempos de ejecución, se visualiza una ligera ventaja para la configuración 2. Aunque ambas configuraciones tienen un mejor promedio para una de las dos ventanas de tiempo, la configuración 2 exhibe una menor desviación estándar en comparación con la configuración 1. Además, la configuración 2 muestra un valor cercano al de la configuración 1 para la ventana de tiempo de 28800 segundos, la cual es la ventana en la que esta configuración obtiene un mejor desempeño.

Tamaño	Ventana[s]	Configuración1		Configuración2	
		Min Solución[s]	Prom. Soluciones[s]	Min Solución[s]	Prom. Soluciones[s]
Pequeño	7200	9406,6	9425,6 ± 9,7	9406,3	9429,9 ± 11,4
	21600	9414,3	9428,4 ± 8,0	9405,6	9428,1 ± 11,3
	28800	9411,6	9429,1 ± 10,7	9424,0	9433,2 ± 6,4
Mediano	7200	7861444,3	8718444,8 ± 442080,7	8038328,9	8891123,7 ± 405952,8
	21600	1216782,6	1378128,7 ± 93745,6	1194726,6	1318898,9 ± 73853,0
	28800	46997,4	50236,3 ± 2406,4	43486,4	49442,4 ± 3038,2
Grande	7200	34985783,8	36070793,3 ± 542395,5	35093043,8	37689937,4 ± 1412400,9
	21600	15548736,1	18488525,6 ± 1689944,1	16866758,5	18814657,1 ± 999027,8
	28800	10531415,0	11623149,7 ± 634930,6	10433793,6	11130082,3 ± 559843,0

Tabla 15: Resumen soluciones: Comuna de Quinta Normal

Tamaño	Ventana[s]	Configuración1		Configuración2	
		Prom. Iteraciones	Prom. Uso	Prom. Iteraciones	Prom. Uso
Pequeño	7200	21,5 ± 16,9	1,32 ± 3,36	25,2 ± 22,8	1,34 ± 5,61
	21600	36,9 ± 47,5	1,33 ± 4,95	25,1 ± 35,5	1,33 ± 4,4
	28800	20,1 ± 24,2	1,33 ± 4,21	22,3 ± 17,5	1,35 ± 3,11
Mediano	7200	32,5 ± 33,9	1,85 ± 24,1	24,9 ± 17,4	1,84 ± 32,5
	21600	47,6 ± 26,5	1,99 ± 66,58	49,7 ± 35,9	2,01 ± 54,01
	28800	33,3 ± 17,7	1,99 ± 46,29	33,4 ± 12,3	1,94 ± 28,94
Grande	7200	41,5 ± 30,5	2,04 ± 25,86	20,8 ± 21,8	2,08 ± 38,28
	21600	36,3 ± 31	2,11 ± 99,97	48,8 ± 42,3	2,04 ± 65,93
	28800	40,7 ± 19,5	2,15 ± 65,73	45,7 ± 38,1	2,08 ± 85,45

Tabla 16: Resumen estadísticas: Comuna de Quinta Normal

Tamaño	Ventana[s]	Configuración1		Configuración2	
		T.E.MinSolución[s]	Prom.T.E.[s]	T.E.MinSolución[s]	Prom.T.E.[s]
Pequeño	7200	117	166,7 ± 39,3	117	175 ± 52,2
	21600	117	202,5 ± 109,7	117	174,8 ± 81,3
	28800	117	163,5 ± 55,8	117	168,3 ± 40,1
Mediano	7200	1423	2307,5 ± 920,9	1392	2082,3 ± 477,3
	21600	1715	2718,3 ± 718,8	1774	2750,1 ± 973,8
	28800	1586	2329,3 ± 479,8	1768	2311,9 ± 331,8
Grande	7200	3191	5889 ± 1984,4	3262	4651,3 ± 1452,4
	21600	3256	5579,3 ± 2034,2	3239	6493,7 ± 2803,7
	28800	4380	5861 ± 1263,4	3298	6277,7 ± 2504

Tabla 17: Resumen tiempos de ejecución: Comuna de Quinta Normal

5.5. Resumen

En este capítulo se presenta una exhaustiva validación del algoritmo desarrollado para el problema del enrutamiento de vehículos de recolección de desechos puerta a puerta con

ventanas de tiempo. El estudio se enfoca en tres aspectos cruciales: evaluación de eficacia, eficiencia y calidad de las soluciones generadas. Se detalla también el meticuloso proceso de sintonización de parámetros mediante un sintonizador que explora combinaciones óptimas para maximizar el rendimiento del algoritmo en diversas situaciones. Además, se describen las instancias de prueba generadas utilizando el formato GeoJSON, específicamente diseñadas para tres comunas urbanas de Chile: Renca, Cerro Navia y Quinta Normal. Estos experimentos proporcionan una comprensión profunda del rendimiento del algoritmo en escenarios del mundo real, fundamentales para la mejora continua y la implementación efectiva del sistema de enrutamiento de vehículos de recolección de desechos, contribuyendo así significativamente al campo de la gestión eficiente de residuos urbanos.

CAPÍTULO 6

CONCLUSIONES

El trabajo presentado propuso un método para resolver el problema del Ruteo de Vehículos de Recolección de Residuos Puerta a Puerta con Ventanas de Tiempo. Este problema implica el enrutamiento de vehículos en donde los arcos deben ser visitados dentro de ciertas ventanas de tiempo. Además, el costo de visitar los arcos varía dependiendo de si han sido o no visitados previamente, esto con el fin de simular así la recolección ya realizada o pendiente. Asimismo, el enrutamiento también está restringido por nodos iniciales y finales, los cuales pueden ser idénticos o distintos, con el propósito de simular el inicio y fin de las rutas desde un vertedero.

El algoritmo propuesto para resolver este problema se basa en la optimización mediante un algoritmo basado en colonias de hormigas. Este algoritmo ha sido modificado para dotar a las hormigas de memoria a corto plazo, permitiéndoles visitar arcos donde la recolección de residuos ya se ha realizado, en caso de que las heurísticas lo consideren apropiado. En cuanto al proceso de generación de soluciones, se divide en dos etapas. La primera consiste en visitar todos los arcos de una instancia desde uno de los nodos iniciales. La segunda se enfoca en encontrar un camino desde el último nodo visitado hasta uno de los nodos finales.

Con respecto a la experimentación del algoritmo, este fue evaluado utilizando zonas de distintos tamaños y con diferentes ventanas de tiempo de tres comunas de Santiago: Renca, Quinta Normal y Cerro Navia, por lo que los casos eran reales. Las instancias de pruebas fueron generadas utilizando un algoritmo desarrollado específicamente para este propósito, lo que permite en un futuro expandir este trabajo a otras comunas o zonas geográficas del país.

En relación a los resultados obtenidos por el algoritmo para las tres comunas, se observa que, en general, este pudo resolver las instancias de tamaño pequeño dentro de las ventanas de tiempo establecidas. No obstante, para casos medianos o grandes, en ciertos casos el algoritmo no logró respetar las ventanas de tiempo, lo que resultó en soluciones no factibles. Sin embargo, es importante destacar que para ciertas instancias de gran tamaño, es comprensible que el algoritmo no haya podido encontrar una solución adecuada, ya que este está limitado a trabajar con un solo camión, por lo que es esperable que este no pueda atender extensas áreas dentro de una comuna. En el futuro se podría modificar este algoritmo para soportar múltiples camiones y así poder resolver instancias más complejas.

Con respecto a la calidad de las soluciones obtenidas, estas se consideran adecuadas. Sin embargo, al tratarse de casos reales, la literatura actual no ofrece resultados comparativos. Por lo tanto, en el futuro sería necesario comparar este algoritmo con otros existentes para obtener una evaluación más completa o con la información real de las comunas.

Continuando con el análisis de la calidad de los resultados, habría sido beneficioso incorporar pasos adicionales en el algoritmo para refinar las rutas seleccionadas por las heurísticas.

Esto se debe a que las hormigas no seleccionaban las rutas más óptimas en algunas ocasiones debido a su limitada visibilidad futura, lo que las llevaba a tomar, en ciertas situaciones, rutas que no eran las más óptimas para llegar a su objetivo. En ocasiones, esto empeoraba con la influencia de las feromonas, lo que limitaba considerablemente las soluciones que las hormigas podían alcanzar. Es por esta razón que en algunas instancias, la configuración 2, la cual limpiaba las feromonas en cada iteración, lograba obtener resultados superiores en comparación con la configuración 1. Sin embargo, la diferencia de desempeño de ambas configuraciones fue leve, lo que implica que las feromonas en cierto sentido no eran un parámetro clave para modificar el comportamiento del algoritmo.

En cuanto a los tiempos de ejecución para las soluciones factibles, se observa que el algoritmo puede encontrar soluciones rápidamente en casos pequeños, pero el tiempo de ejecución aumenta significativamente para instancias de tamaño mediano o grande. En el caso de las soluciones infactibles, se presenta una tendencia similar, con tiempos de ejecución comparables a los de los casos factibles. No obstante, podría ser beneficioso contemplar la aplicación de un límite de tiempo en el proceso, al igual que se incorporó un parámetro para controlar la cantidad de iteraciones sin mejora.

En cuanto a las configuraciones seleccionadas por el sintonizador, es posible que estas no hayan sido las de mejor desempeño debido a que resultaron ser muy extremas para las instancias que se evaluaban. Por lo tanto, las 500 o 1000 iteraciones que se establecieron para las configuraciones 1 y 2 respectivamente, no se alcanzaron nunca durante la experimentación, ni siquiera para los casos más grandes. Esto podría deberse a que el algoritmo encontró la mejor solución posible para la instancia en una cantidad menor de iteraciones o, simplemente, no pudo encontrar una mejora significativa debido a que quedó atrapado en óptimos locales. Así, después de 50 iteraciones sin mejoras, el algoritmo finalizaba, lo que implica que finalmente solo se comparó el coeficiente de evaporación entre ambas configuraciones.

Con respecto a la segunda fase del algoritmo, la cual era la encargada de buscar una ruta desde el ultimo nodo recorrido hasta uno de los nodos finales, podría haber sido beneficioso incorporar una nueva técnica solo para este proceso, como lo es el algoritmo A^* [Foead *et al.*, 2021]. Esto, debido a que en ciertas instancias las soluciones obtenidas por el algoritmo eran infladas hasta un 20 % solamente por el camino de vuelta lo que provocaba que las soluciones hayan sido peores de lo que realmente pudieron haber sido y además, este proceso aumentaba el tiempo de ejecución del algoritmo y el uso de las calles en sus soluciones.

En relación a lo mencionado anteriormente, habría sido beneficioso también incorporar una heurística adicional en el algoritmo la cual permitiera modificar el costo de los arcos dependiendo de su cercanía a los nodos finales para que, a medida que las hormigas recorren todas los arcos, estas tiendan a terminar mas cerca del nodo final. Esta adición habría contribuido a optimizar la segunda fase del algoritmo, permitiendo una mayor eficiencia en la selección de rutas por parte de las hormigas.

Finalmente, los resultados indican que la técnica propuesta en el estudio es viable para abor-

dar este tipo de problema, ya que logra cumplir con los objetivos establecidos. No obstante, como se mencionó previamente en este capítulo, es crucial resaltar que hay margen para mejoras en diversas áreas del algoritmo, lo que podría ser abordado en investigaciones futuras.

REFERENCIAS BIBLIOGRÁFICAS

- [Archetti *et al.*, 2014] Archetti, C., Corberán, A., Plana, I., Sanchis, J. M., y Speranza, M. (2014). The team orienteering arc routing problem. *Transportation Science*.
- [Babaei Tirkolaee *et al.*, 2016] Babaei Tirkolaee, E., Alinaghian, M., Bakhshi, M., y Esfahani, M. (2016). Solving a robust capacitated arc routing problem using a hybrid simulated annealing algorithm: A waste collection application.
- [Belenguer y Benavent, 2003] Belenguer, J. M. y Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30(5):705–728.
- [Benavent *et al.*, 2005] Benavent, E., Corberán, A., Piñana, E., Plana, I., y Sanchis, J. M. (2005). New heuristic algorithms for the windy rural postman problem. *Computers Operations Research*, 32(12):3111–3128.
- [Benavent *et al.*, 2014] Benavent, E., Corberán, A., Desaulniers, G., Lessard, F., Plana, I., y Sanchis, J. M. (2014). A branch-price-and-cut algorithm for the min-max k-vehicle windy rural postman problem. *Networks*, 63.
- [Benavent *et al.*, 2019] Benavent, E., Ángel Corberán, Laganà, D., y Vocaturro, F. (2019). The periodic rural postman problem with irregular services on mixed graphs. *European Journal of Operational Research*, 276(3):826–839.
- [Bianchessi *et al.*, 2022a] Bianchessi, N., Ángel Corberán, Plana, I., Reula, M., y Sanchis, J. M. (2022a). The min-max close-enough arc routing problem. *European Journal of Operational Research*, 300(3):837–851.
- [Bianchessi *et al.*, 2022b] Bianchessi, N., Ángel Corberán, Plana, I., Reula, M., y Sanchis, J. M. (2022b). The profitable close-enough arc routing problem. *Computers Operations Research*, 140:105653.
- [Blazquez y Paredes-Belmar, 2020] Blazquez, C. y Paredes-Belmar, G. (2020). Network design of a household waste collection system: A case study of the commune of renca in santiago, chile. *Waste Management*, 116:179–189.
- [Bošković *et al.*, 2016] Bošković, G., Jovicic, N., Jovanović, S., y Simovic, V. (2016). Calculating the costs of waste collection: A methodological proposal. *Waste Management & Research*, 34.
- [Braier *et al.*, 2017] Braier, G., Durán, G., Marengo, J., y Wesner, F. (2017). An integer programming approach to a real-world recyclable waste collection problem in argentina. *Waste Management & Research*, 35(5):525–533. PMID: 28462677.
- [Butler *et al.*, 2016] Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., y Schaub, T. (2016). The geojson format. Technical report.

- [Campbell *et al.*, 2021] Campbell, J. F., Ángel Corberán, Plana, I., Sanchis, J. M., y Segura, P. (2021). Solving the length constrained k-drones rural postman problem. *European Journal of Operational Research*, 292(1):60–72.
- [Carr y Simonetti, 2020] Carr, R. D. y Simonetti, N. (2020). A new integer programming formulation of the graphical traveling salesman problem.
- [Colombi *et al.*, 2017] Colombi, M., Ángel Corberán, Mansini, R., Plana, I., y Sanchis, J. M. (2017). The hierarchical mixed rural postman problem: Polyhedral analysis and a branch-and-cut algorithm. *European Journal of Operational Research*, 257(1):1–12.
- [Corberán *et al.*, 2005] Corberán, A., Mejía, G., y Sanchis, J. M. (2005). New results on the mixed general routing problem. *Operations Research*, 53(2):363–376.
- [Corberán *et al.*, 2019] Corberán, Á., Plana, I., Reula, M., y Sanchis, J. M. (2019). A matheuristic for the distance-constrained close-enough arc routing problem. *TOP*, 27(2):312–326.
- [Corberán *et al.*, 2000] Corberán, A., Martí, R., y Romero, A. (2000). Heuristics for the mixed rural postman problem. *Computers Operations Research*, 27(2):183–203.
- [Corberán *et al.*, 2002] Corberán, A., Martí, R., y Sanchis, J. (2002). A grasp heuristic for the mixed chinese postman problem. *European Journal of Operational Research*, 142(1):70–80.
- [Corberán *et al.*, 2007] Corberán, A., Plana, I., y Sanchis, J. M. (2007). A branch & cut algorithm for the windy general routing problem and special cases. *Networks*, 49(4):245–257.
- [Dorigo *et al.*, 1996] Dorigo, M., Maniezzo, V., y Coloni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.
- [Feo y Resende, 1995] Feo, T. A. y Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.
- [Foad *et al.*, 2021] Foad, D., Ghifari, A., Kusuma, M. B., Hanafiah, N., y Gunawan, E. (2021). A systematic literature review of a* pathfinding. *Procedia Computer Science*, 179:507–514.
- [Frederickson, 1979] Frederickson, G. N. (1979). Approximation algorithms for some postman problems. *J. ACM*, 26(3):538–554.
- [Frederickson *et al.*, 1976] Frederickson, G. N., Hecht, M. S., y Kim, C. E. (1976). Approximation algorithms for some routing problems. En *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pp. 216–227.
- [Gordenko y Avdoshin, 2018] Gordenko, M. y Avdoshin, S. (2018). The variants of chinese postman problems and way of solving through transformation into vehicle routing problems. *Proceedings of the Institute for System Programming of the RAS*, 30:221–232.

- [Groves y Vuuren, 2005] Groves, G. y Vuuren, J. (2005). Efficient heuristics for the rural postman problem. *ORiON*, 21.
- [Guan, 1984] Guan, M. (1984). On the windy postman problem. *Discrete Applied Mathematics*, 9(1):41–46.
- [Hertz et al., 2000] Hertz, A., Laporte, G., y Mittaz, M. (2000). A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1):129–135.
- [Hutter et al., 2009] Hutter, F., Hoos, H. H., Leyton-Brown, K., y Stützle, T. (2009). Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306.
- [Hà et al., 2014] Hà, M. H., Bostel, N., Langevin, A., y Rousseau, L.-M. (2014). Solving the close-enough arc routing problem. *Networks*, 63(1):107–118.
- [Lacomme et al., 2001] Lacomme, P., Prins, C., y Ramdane-Chérif, W. (2001). A genetic algorithm for the capacitated arc routing problem and its extensions. En Boers, E. J. W., editor, *Applications of Evolutionary Computing*, pp. 473–483, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Letchford y Eglese, 2001] Letchford, A. y Eglese, R. (2001). *The General Routing Problem*, pp. 206–207.
- [Li y Huang, 2010] Li, Y. y Huang, G. (2010). Modeling municipal solid waste management system under uncertainty. *Journal of the Air & Waste Management Association*, 60(4):439–453.
- [Nobert y Picard, 1996] Nobert, Y. y Picard, J.-C. (1996). An optimal algorithm for the mixed chinese postman problem. *Networks*, 27(2):95–108.
- [Pearn y Wu, 1995] Pearn, W. y Wu, T. (1995). Algorithms for the rural postman problem. *Computers Operations Research*, 22(8):819–828.
- [Pearn et al., 1987] Pearn, W.-L., Assad, A., y Golden, B. L. (1987). Transforming arc routing into node routing problems. *Computers & Operations Research*, 14(4):285–288.
- [Pearn y Chiu, 2005] Pearn, W. L. y Chiu, W. C. (2005). Approximate solutions for the maximum benefit chinese postman problem. *International Journal of Systems Science*, 36(13):815–822.
- [Ralphs, 1993] Ralphs, T. (1993). On the mixed chinese postman problem. *Operations Research Letters*, 14(3):123–127.
- [Shaw, 1998] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. En Maher, M. y Puget, J.-F., editores, *Principles and Practice of Constraint Programming — CP98*, pp. 417–431, Berlin, Heidelberg. Springer Berlin Heidelberg.

- [Ávila *et al.*, 2015] Ávila, T., Corberán, A., Plana, I., y Sanchis, J. M. (2015). The stacker crane problem and the directed general routing problem. *Networks*, 65(1):43–55.
- [Ávila *et al.*, 2016] Ávila, T., Ángel Corberán, Plana, I., y Sanchis, J. M. (2016). A branch-and-cut algorithm for the profitable windy rural postman problem. *European Journal of Operational Research*, 249(3):1092–1101.