UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

Optimización de rutas de recolección utilizando colonia de hormigas para un problema de ruteo por arcos

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

FRANCISCO JAVIER MÁRQUEZ CORTÉS

DIRECTOR DE TESIS

DRA. BEATRIZ AURORA GARRO LICÓN

CIUDAD UNIVERSITARIA, CD. MX. 2019

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

1. Datos del alumno
Márquez
Cortés
Francisco Javier
26 43 06 00
Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
311279708

2. Datos del tutor
Dra.
Beatriz Aurora
Garro
Licón

3. Datos del sinodal 1
Dra.
Katya
Rodríguez
Vázquez

4. Datos del sinodal 2
Dr.
José David
Flores
Peñaloza

5. Datos del sinodal 3
Dr.
José de Jesús
Galaviz
Casas

6. Datos del sinodal 4
Dra.
María de Luz
Gasca
Soto

7. Datos del trabajo escrito
Optimización de rutas de recolección utilizando colonia de hormigas para un problema de ruteo por arcos
87 p
2019

Agradecimientos

A la SECITI por el proyecto “Optimización de Rutas para Recolección de Residuos Sólidos en las Delegaciones de la Ciudad de México” con número SECITI/064/2016 y el apoyo económico brindado para la realización del mismo.

A la doctora Beatriz Aurora Garro Licón por su apoyo y paciencia a lo largo del trabajo y a la doctora Katya Rodríguez Vázquez por su amabilidad y su experiencia que ayudó a la mejora de este proyecto. Estoy agradecido de haber podido trabajar con ustedes.

A mis padres, por el apoyo incondicional y la motivación constante para seguir adelante. Sé lo importante que es para ellos este logro y espero que esto sea una muestra de que todo el esfuerzo que dedicaron a mi educación no fue en vano.

A la familia Reyes Avendaño, por su amabilidad, apoyo y sugerencias para este trabajo.

A Karen Gill, quien ha estado a mi lado en los mejores y peores momentos y que sin su ayuda, quizá no habría concluido este trabajo. Sin duda hacemos un gran equipo.

Resumen

En esta tesis se plantea una propuesta de solución a la problemática de la generación de rutas para la recolección de residuos utilizando una variante de un algoritmo bioinspirado llamado **Optimización por Colonia de Hormigas**. Como caso de estudio para el desarrollo de este trabajo, se eligió a la colonia Villa Milpa Alta de la delegación [1] Milpa Alta. Para esto, se hizo una transformación del problema real dentro de la colonia a una instancia del **problema de ruteo por arcos con capacidad** haciendo uso de la teoría de grafos. Al mismo tiempo, se realizó la implementación de la colonia de hormigas y se evaluó su desempeño utilizando un conjunto de *benchmarks*. Como resultado se obtuvo un programa que permitía generar rutas de recolección no solo sobre la colonia Villa Milpa Alta, sino para cualquier zona geográfica.

[1] A partir del 1 de octubre del año 2018 se aplicó una reforma política en la Ciudad de México la cual incluye, entre otros cambios, la transición de delegación a alcaldía. En este trabajo se utilizará el término delegación ya que éste se inició previo a la aplicación de la reforma antes mencionada.

Índice general

Índice de figuras

Índice de tablas

Glosario

ACO Ant Colony Optimization, metaheurística inspirada en el comportamiento cooperativo de las hormigas. 16, 17, 29, 38, 41–43, 57, 75

ACS Ant Colony System, una de las variantes de los algoritmos de colonia de hormigas. 44, 45

ARP Arc Routing Problem, problema de ruteo por arcos. 6, 34–36, 38, 39

AS Ant System, primer variante de los algoritmos de colonia de hormigas. 29

CARP Capacitated Arc Routing Problem, problema de ruteo por arcos con capacidad. 36–39, 58, 66, 75

CCPP Capacitated Chinese Postman Problem, problema del cartero chino con capacidad. 36

CPP Chinese Postman Problem, problema del cartero chino. 34

CVRP Capacitated Vehicle Routing Problem, problema de ruteo de vehículos con capacidad. 38

DCARP Directed Capacitated Arc Routing Problem, problema de ruteo por arcos dirigidos con capacidad. 36

ECARP Extended Capacitated Arc Routing Problem, problema extendido de ruteo por arcos con capacidad. 38

HACOA Hybrid Ant Colony Optimization Algorithm. 38

HCPP Hierarchy Chinese Postman Problem, problema del cartero chino con jerarquías. 36

MBCPP Maximum Benefit Chinese Postman Problem, problema del cartero chino con máximo beneficio. 36

MCPP Mixed Chinese Postman Problem, problema del cartero chino mixto. 35

MMAS MMAS, variante de los algoritmos de colonia de hormigas. 38, 43

NEARP Node, Edge and Arc Routing Problem, problema de ruteo por arcos, aristas y nodos. 39

PARPRP Periodic Capacitated Arc Routing Problem with Refill Points, problema periódico de ruteo por arcos con capacidad y puntos de recarga. 42

PCARP Periodic Capacitated Arc Routing Problem, problema periódico de ruteo por arcos con capacidad. 39

RPP Rural Postman Problem, problema del cartero rural. 37, 38

SA Simulated Annealing, recocido simulado, metaheurística inspirada en el proceso de recocido utilizado al forjar metales y cerámicas. 45

TSP Travelling Salesman Problem, problema del agente viajero. 36, 42, 43

UCARP Undirected Capacitated Arc Routing Problem, problema de ruteo por arcos no dirigidos con capacidad. 36

VRP Vehicle Routing Problem, problema de ruteo de vehículos. 36, 39, 43

VRPSD Vehicle Routing Problem with Stochastic Demand, problema de ruteo de vehículos con demanda estocástica. 44

WPP Windy Postman Problem, problema del cartero ventoso. 35

Introducción

La contaminación es una de las problemáticas más grandes alrededor del mundo. En ciudades densamente pobladas como la Ciudad de México, la generación de residuos sólidos es uno de los principales factores contaminantes. Actualmente, se producen alrededor de 12,893 toneladas de basura por día por lo que es muy importante contar con los medios y técnicas adecuadas para poder disponer de todos esos residuos de manera adecuada.

El manejo de la basura generada por las viviendas en la ciudad, se ve afectado por otras problemáticas como tráfico, falta de personal, vialidades bloqueadas o en reparación, entre otras. Además, en México se tiene la costumbre de que el camión recolector pase frente a cada hogar para llevarse los residuos, por lo que sería difícil que los usuarios acepten otra forma de deshacerse de su basura. Debido a esto se deben de buscar nuevas técnicas que ayuden a mejorar esta forma de recolección sin afectar a los ciudadanos, y además, que reduzca los tiempos de traslado así como la distancia que se recorre con respecto a las rutas que se utilizan de forma cotidiana.

Hablando de movilidad, en la Ciudad de México, no es fácil encontrar una buena ruta para desplazarse de un lugar a otro ya que continuamente enfrentamos problemas como manifestaciones, inundaciones, obras, fiestas y celebraciones, embotellamientos, entre otros. Esto no solo incrementa los tiempos de traslado, sino que aumenta los gastos de operaciones y cambia constantemente las mejores rutas para transitar. Por lo que tener una herramienta que auxilie en la búsqueda de rutas en determinados momentos, sería de gran ayuda para los camiones recolectores que tienen que lidiar con estos problemas diariamente para poder proporcionar el servicio de recolección a todos los ciudadanos.

La ruta que siguen los camiones para recoger los residuos ha sido obtenida de manera empírica por los mismos operadores. Es por ello que aplicar técnicas computacionales a este problema puede mejorar el manejo de la recolección de residuos, disminuir costos de operación y obtener un conjunto de rutas posibles a seguir que serán las mejores bajo ciertas condiciones.

Motivación

En el año de 1992 Marco Dorigo propuso una técnica para obtener una ruta óptima en un grafo [14]; dicha técnica estaba basada en el comportamiento de las hormigas cuando están en busca de alimento. Esta técnica sería conocida como Optimización por colonia de hormigas (ACO por sus siglas en inglés). A partir de su publicación han aparecido bastantes soluciones a problemas en grafos, las cuales han sido obtenidas utilizando esta técnica. Uno de estos problemas es el llamado "problema del cartero chino con capacidad" el cual nos pide encontrar una ruta que minimice distancia para que un grupo de carteros con cierta capacidad de carga entregue la correspondencia en todas las calles de una ciudad. Al ser un problema NP-Duro no tiene algoritmo que pueda resolverlo en tiempo polinomial, pero utilizando ACO, podemos obtener una muy buena solución en relativamente poco tiempo.

Tratando de aplicar todo lo anterior a un problema real, por ejemplo, la recolección de residuos donde un camión transita por las calles recolectando desechos sin dejar una calle sin servicio, lo podemos tratar como el problema del cartero chino con capacidad. Además vale la pena intentar mejorar este servicio, sobre todo en la Ciudad de México, por la importancia del cuidado del ambiente, por la creciente demanda de servicios debido al incremento de la población y porque la acumulación de basura representa un problema de salud pública por lo que, la motivación principal de este trabajo consiste en mejorar las rutas de recolección de residuos sólidos en uno de los doce pueblos pertenecientes a la delegación Milpa Alta (Villa Milpa Alta) con el fin de reducir distancias y tiempo de traslado a los camiones recolectores y mejorar el servicio para beneficio de la población en general.

Problema a resolver

El problema a resolver consiste en optimizar la distancia de las rutas que pueda seguir un vehículo en un proceso de recolección de residuos.

Objetivo General

El objetivo de este trabajo es transformar el problema de recolección de residuos a una instancia del problema de ruteo por arcos (ARP por sus siglas en inglés) en un grafo, para poder aplicar una técnica de optimazación por colonia de hormigas y optimizar la distancia de las rutas. Lo que se va a realizar en este trabajo se puede resumir a cuatro puntos importantes:

- Obtener el grafo correspondiente a la zona geográfica sobre la que se desea trabajar (en este caso, las calles de la colonia Villa Milpa Alta).
- Hacer alguna transformación, en caso de ser necesario, para obtener una instancia del problema de ruteo por arcos, es decir, un grafo que represente mejor la zona y sus restricciones.
- Aplicar la técnica de ACO a la instancia obtenida en el punto anterior para obtener nuevas rutas.
- Optimizar el programa para asegurar el buen funcionamiento del mismo.

Restricciones

El desarrollo de este trabajo se lleva a cabo siguiendo una idea de tener una representación gráfica del problema real para poder así aplicar las técnicas descritas anteriormente sin problema alguno. Estas son las restricciones del problema a resolver.

- Solo se va a trabajar en la colonia Villa Milpa Alta.
- El problema no va a ser dinámico, es decir, las actualizaciones al mapa no serán en tiempo real.
- No cubre eventos aleatorios causados por un fenómeno natural o social (estado del tiempo, eventos culturales, accidentes viales, obras, etcétera.)
- Los cálculos se realizarán con valores aproximados con base en datos estadísticos para que sean lo más congruentes posibles con la situación real de la colonia.

Aportaciones

A continuación se enlistan las aportaciones:

- Este trabajo resuelve una problemática social real poco tratada hasta ahora.
- Mejorar las rutas que se utilizan actualmente para disponer de los residuos sólidos en la delegación Milpa Alta, tomando como muestra representativa la colonia Villa Milpa Alta.
- Minimizar la distancia total de recorrido para cada camión.
- Mejorar el servicio de recolección para un beneficio directo en la sociedad.

- Sentar un precedente en la aplicación de estas técnicas a problemáticas similares que se presentan en la ciudad.

Capítulo 1

Conceptos básicos

En este capítulo, se definirán los conceptos teóricos necesarios para entender la base de esta investigación, para comprender la metodología y los resultados obtenidos. Los conceptos esenciales de este capitulo son: un acercamiento a la teoría de grafos, la definición del problema de ruteo de vehículos y algunas variantes así como la definición de optimización y diferentes algoritmos basados en la optimización por colonia de hormigas.

1.1 Teoría de grafos[1]

De manera sencilla, podemos ver a un **grafo** como un conjunto de puntos y otro de líneas, las cuales conectan a los puntos entre sí dada una relación común entre ellos. Los puntos reciben el nombre de *vértices* o *nodos* y las líneas se llaman *adyacencias*. Los vértices pueden representar cualquier objeto o entidad como personas, ciudades, números o estaciones en una red; mientras que las adyacencias representan las relaciones que existen entre ellos, como pueden ser la amistad entre un conjunto de personas, los diferentes caminos que puede haber entre un conjunto de ciudades, la relación matemática entre dos o más valores, etcétera. Para representar la relación entre dos vértices cualesquiera, normalmente se utilizan tuplas de la forma (a,b) donde a y b son vértices; esto se puede leer como "a está relacionado con b".

Formalmente, un grafo G se representa como $G = (V,E)$ donde V es un conjunto de elementos llamados vértices y E es un subconjunto de elementos del producto cartesiano $V \times V$, de modo que $E \subseteq V \times V$ y representa las adyacencias. Cada vértice presenta un valor asociado llamado *grado*, el cuál indica la cantidad de adyacencias

[1] El material de esta sección está basado principalmente en el libro de Berge [6].

que inciden en él. Como las adyacencias representan la relación entre cualesquiera dos vértices a, b, puede suceder cualquiera de los siguientes casos:

- a está relacionado con b pero b no está relacionado con a.
- a está relacionado con b y, al mismo tiempo, b está relacionado con a.
- a está relacionado con b pero $b = a$.

Dado las relaciones anteriores se tienen tres tipos de adyacencias: los arcos, las aristas y los lazos.

Un **arco** es una adyacencia en la cuál la dirección de la relación está definida de manera explícita. Por esto, un arco representa el primer caso. Para verlo de manera gráfica se utiliza una flecha que indica el sentido de la relación.

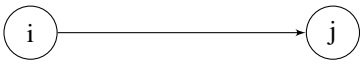

Figura 1.1: Ejemplo de arco

Siguiendo la definición de Berge, una **arista** representa el segundo caso, e indica que la relación se da en ambos sentidos ya que no hay dirección explícita. Para representarla, se utiliza una simple línea o una flecha bidireccional.

Figura 1.2: Ejemplos de arista

Berg define a un **lazo** (también le llama *bucle*) como la relación que tiene un vértice consigo mismo. Se utiliza una flecha o línea que empieza y termina en el mismo vértice para representarlo.

Figura 1.3: Ejemplos de lazo

1.1.1 Tipos de grafos

Existe una clasificación de grafos de acuerdo al tipo de adyacencias que presentan; los **grafos dirigidos** son aquellos en los que las relaciones entre vértices están denotadas por arcos, es decir, todas tienen dirección específica como lo muestra la Figura 1.4. Los **grafos no dirigidos**, por el contrario, no presentan dirección en ninguna de sus adyacencias, es decir, todas son aristas como en el ejemplo de la Figura 1.5. Por último, tenemos los **grafos mixtos** que, como su nombre lo indica, presentan arcos y aristas por igual; esto se ilustra en la Figura 1.6. Las adyacencias a su vez, pueden tener un valor asociado llamado *peso*, en caso de no indicar un peso, se tomará el valor de 1.

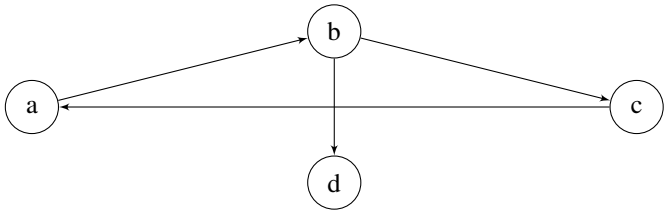

Figura 1.4: Ejemplo de grafo dirigido

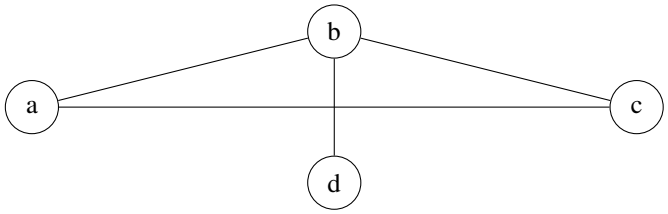

Figura 1.5: Ejemplo de grafo no dirigido

Para ejemplificar los conceptos anteriores imaginemos a cuatro personas, las cuales vamos a representar con las letras A, B, C y D. Supongamos que en este grupo de personas la relación de conocerse se da de acuerdo a las siguientes reglas:

- A conoce a B y C.
- B conoce a A, C y D.
- C conoce a A y B.

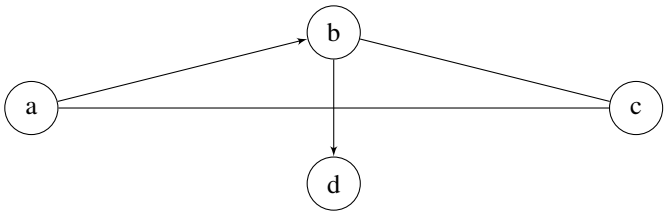

Figura 1.6: Ejemplo de grafo mixto

- D sólo conoce a B.

A partir de esta información, podemos construir nuestro conjunto de vértices V de la siguiente manera: $V = \{A, B, C, D\}$ y la relación de conocerse la podemos escribir como un conjunto de parejas ordenadas E, lo que daría lugar al conjunto de aristas $E = \{(A,B),(A,C),(B,A),(B,C),(B,D),(C,A),(C,B),(D,B)\}$. Con base en lo anterior, es posible representar un grafo $G = (V,E)$ de la siguiente manera:

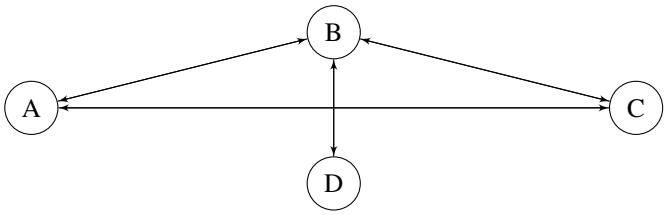

Figura 1.7: Ejemplo de grafo

Algunas definiciones importantes que son básicas y necesarias en teoría de grafos son las siguientes [23]:
Sean x, y vértices (no necesariamente distintos) de un grafo no dirigido $G = (V,E)$.

1. Un *camino x-y* en G es una sucesión alternada finita sin lazos W tal que $W = \{x = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k = y\}$, cuyos términos son vértices y adyacencias de manera alternada de modo que, $e_i = (v_{i-1}, v_i)$ con $1 \leq i \leq k$.

2. La *longitud* de un camino es k, donde k es el número de adyacencias que contiene el camino. Si $k = 0$ entonces es un camino sin aristas y $x = y$. A este camino se le denomina trivial.

3. Cualquier camino donde $x = y$ es un *camino cerrado*. Esto quiere decir que empieza y termina en el mismo vértice, si no sucede así, el camino es *abierto*.

4. Sea W un camino $x - y$ en un grafo G, si no se repite ninguna adyacencia en W entonces el camino es un *recorrido* $x - y$, también conocido como trayectoria.

5. Un recorrido $x - x$ cerrado es un *circuito*.

6. Si ningún vértice del camino $x - y$ se presenta más de una vez, el camino es un *camino simple x-y*.

7. Un *ciclo* consiste en un camino simple $x - x$.

Las definiciones anteriores pueden verse resumidas en la Tabla 1.1.

Tabla 1.1: Recorridos en grafos

Vértices repetidos	**Aristas repetidas**	**Abierto**	**Nombre**
Sí	Sí	Sí	Camino
Sí	Sí	No	Camino cerrado
Sí	No	Sí	Recorrido
Sí	No	No	Circuito
No	No	Sí	Camino simple
No	No	No	Ciclo

1.1.2 Recorridos en grafos

Una vez que se tiene la representación de un grafo podemos recorrerlo a través de sus aristas. Existen diversos nombres para la forma y las restricciones de cada recorrido, los más famosos son el circuito Euleriano y el ciclo Hamiltoniano.

El problema de los puentes de Könisberg fue un pequeño desafío que surgió entre los habitantes de la ciudad. Könisberg era atravesada por el río Pregol, por lo que existían distintas zonas conectadas por medio de puentes (La Figura 1.8 presenta un dibujo del mapa de los puentes de Könisberg). El problema consiste en saber si una persona puede cruzar todos los puentes una sola vez, regresando al lugar del cual partió. Euler utilizó una aproximación de teoría de grafos para modelar el problema, siendo su objetivo encontrar un circuito que recorre todas las aristas una sola vez, partiendo y regresando al mismo vértice. Euler dio solución al problema en el año de 1736.

Debido a que la teoría de grafos surge gracias a la solución que Euler dio al problema de los puentes de Königsberg, se le denomina circuito Euleriano a aquel circuito que recorra las aristas de un grafo G exactamente una vez. También existe el recorrido Euleriano, el cual consiste en un recorrido x, y que contenga todas las aristas de un grafo exactamente una vez, es decir, que no se repitan aristas. Gracias al trabajo de Euler tenemos una caracterización para decidir si un grafo tiene un circuito Euleriano o no; esta caracterización consiste en evaluar el grado de cada vértice del grafo, si todos los vértices tienen un grado par mayor a 1, entonces existe un circuito Euleriano [23].

Figura 1.8: Mapa de los puentes

El problema del cartero chino es un problema famoso en teoría de grafos e investigación de operaciones y consiste en encontrar un circuito Euleriano de peso mínimo. Se retomará en la sección de problemas de ruteo.

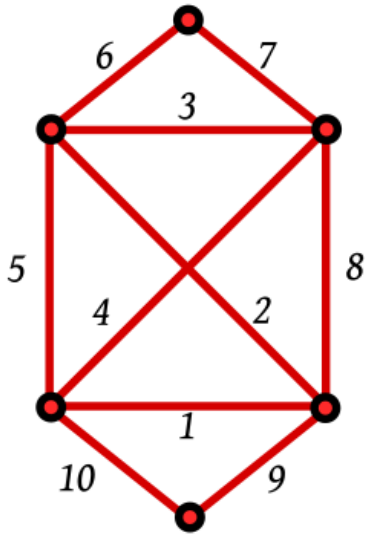

Figura 1.9: Circuito Euleriano

El segundo recorrido más famoso es el ciclo Hamiltoniano, el cual consiste en recorrer todos los vértices exactamente una vez [41] y de igual manera, tenemos el recorrido Hamiltoniano que consiste en un recorrido x, y que contenga todos los vértices de un grafo exactamente una vez. Se llama ciclo Hamiltoniano en honor a William Rowan Hamilton, creador de un juego que consistía en encontrar un ciclo Hamiltoniano en las aristas de un dodecaedro. La búsqueda de un ciclo Hamiltoniano es mucho más difícil que la búsqueda de un circuito Euleriano, de hecho, el problema del ciclo Ha-

miltoniano se encuentra en la categoría de los problemas NP-Completos [26], de modo que la mejor forma que tenemos hasta ahora para encontrarlo es realizar una búsqueda exhaustiva sobre todas las opciones. Entre los problemas más famosos dentro de la teoría de grafos se encuentra el problema del agente viajero, que consiste en encontrar un ciclo Hamiltoniano de peso mínimo. Este problema será retomado más adelante.

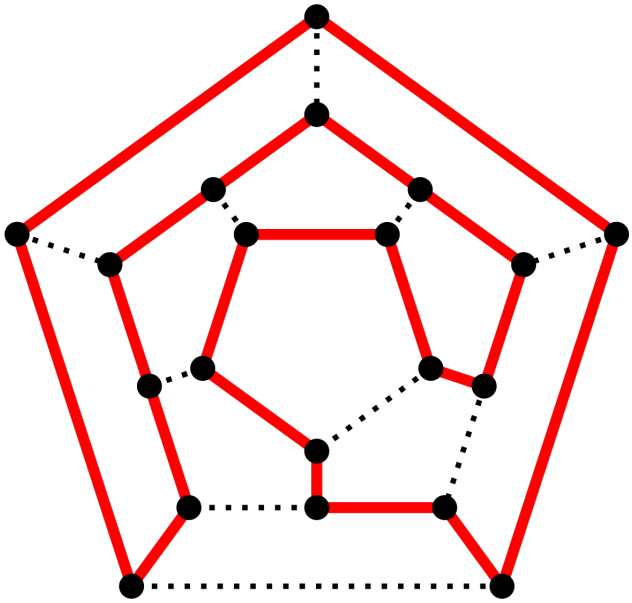

Figura 1.10: Ciclo Hamiltoniano

Estos conceptos son clave para el desarrollo de los problemas de ruteo que, en este trabajo, tienen mucha relevancia debido a que permiten entender el planteamiento, desarrollo y solución del caso que se presenta.

1.2 Optimización

Cuando se toma una decisión siempre buscamos que sea la mejor, pero, ¿cómo sabemos que es realmente la mejor? Para tener certeza en esto, tendríamos que conocer todas las posibles opciones y compararlas una a una para determinar si realmente es la mejor o no. Este proceso al cual todos nos hemos enfrentado al menos una vez en la vida se le llama **optimización**.

Algunas definiciones de este término son: La optimización es el acto de obtener el mejor resultado posible bajo ciertas circunstancias [3]. Es el proceso de determinar la

mejor opción [40]. La optimización es una técnica ampliamente usada en la Investigación de Operaciones que ha sido empleada en muchas aplicaciones. El objetivo es maximizar o minimizar una función sujeta a un conjunto de restricciones [43].

Cuando nos encontramos en una situación en la cual tenemos varias opciones posibles, un conjunto de restricciones y una forma de evaluar las diversas opciones para conocer su calidad, estamos frente a un problema de optimización. Los problemas de optimización combinatoria son aquellos cuya finalidad es encontrar el mejor estado de acuerdo a una función objetivo y constan de dos componentes principales: el problema, que es la cuestión en general por resolver y un conjunto de instancias, que son problemas con valores específicos para los parámetros [47].

Sea π un problema de optimización, una instancia de π es una tripleta (S, f, Ω) donde S es el conjunto de soluciones candidatas, f es la función objetivo que asigna un valor $f(s)$ a cada $s \in S$, y Ω es el conjunto de restricciones. Las soluciones que satisfacen a todas las restricciones en Ω se llaman *soluciones factibles* y se representan como el conjunto $\widetilde{S} \subseteq S$. El objetivo es encontrar una solución factible $s^* \in S$ de modo que, si el problema es de minimización, $f(s^*) \leq f(s) \quad \forall s \in \widetilde{S}$ [14]. Un ejemplo podría ser encontrar al máximo global de una función matemática $f : \mathbb{N} \to \mathbb{N}$, el conjunto de posibles soluciones corresponde al dominio de la función $\mathbb{N}$, pero encontrar el valor que maximiza la función requiere evaluar todos los posibles elementos del dominio o aplicar técnicas de cálculo.

Cuando se dice que un problema es de *optimización combinatoria*, se refiere al hecho de que el dominio de la función objetivo es finito o es numerable. A los problemas de optimización combinatoria también se les conoce como problemas de optimización discreta. Algunos de los problemas de optimización combinatoria más famosos en la computación son: problema del agente viajero [7], problema del cartero chino [29], problema de la mochila [36] y coloración de grafos [52].

Para resolver este tipo de problemas utilizando una computadora se aplican varias técnicas, una de ellas son los algoritmos evolutivos y otra son las funciones heurísticas, las cuales se explicarán más a detalle en la siguiente sección, haciendo énfasis en los algoritmos evolutivos.

1.2.1 Métodos de solución

De acuerdo a Russell y Norvig, una **función heurística** $h(n)$ estima el costo de una solución a partir de un estado n [47]. Decimos que una heurística es la aplicación de toda la información que tengamos de una instancia de un problema que nos permite encontrar una solución aproximada en menos tiempo del necesario para encontrar una solución óptima, al ser creada para una instancia en particular, no garantiza su efectividad en otras instancias, además, la calidad de la solución es variable y puede ser muy buena en algunos casos y mala en otros.

Un ejemplo de heurística muy común en la vida diaria es el de encontrar el camino a algún lugar que nunca hemos visitado, nos guiamos por la sensación de estar cerca de nuestro destino y en general funciona, además nos evita el hecho de buscar sobre todas las formas posibles de llegar a dicho lugar pero no nos dice si nuestra ruta fue la mejor o que tan lejos de la solución óptima estamos (y la mayoría de las veces no nos importa). Es importante destacar que una heurística no es un algoritmo ya que al ejecutarla sobre la misma instancia más de una vez, puede darnos resultados distintos ya que están diseñadas para encontrar una solución en un tiempo razonable, no para encontrar la solución óptima.

Una vez que hemos definido una heurística, nos damos cuenta de que es muy difícil el tener una heurística distinta para cada problema que se nos presente, pero también, podemos ver que algunas heurísticas las podemos reusar en otros problemas, ya que su planteamiento no tiene nada que ver con la instancia del problema en sí. De esta manera surgen las **metaheurísticas**, que se definen como: un proceso de generación iterativa que guía a una heurística subordinada al combinar de manera inteligente distintos conceptos de exploración y aprovechamiento del espacio de búsqueda, se utilizan estrategias de aprendizaje para estructurar la información a modo de encontrar soluciones cercanas a la óptima [38]. Otra definición de metaheurística es: el conjunto de conceptos algorítmicos que pueden ser usados para definir métodos heurísticos aplicables a un conjunto de problemas [14].

Existen varios ejemplos de metaheuristicas, entre los más famosos se encuentran los **algoritmos evolutivos**, los cuales son un método de búsqueda probabilísta que actúa sobre una población de soluciones simulando, a un nivel de abstracción alto, la evolución de las especies en la naturaleza [42]. Entre los algoritmos evolutivos podemos encontrar a los algoritmos genéticos [24], programación genética [28] y colonia de hormigas [14]. Todas estas técnicas han sido aplicadas a una gran diversidad de problemas de manera satisfactoria porque no fueron creadas pensando en resolver un problema en particular, sino más bien como una herramienta que permitiera optimizar la codificación recibida, por lo que basta poder transformar algún problema a la forma requerida por alguna de estas técnicas para utilizarla.

1.3 Optimización por colonia de hormigas

Las hormigas son uno de los insectos que más han llamado la atención de las personas debido a su comportamiento social. Una de las habilidades más sorprendentes que tienen es la de encontrar un camino de su hormiguero hacia alguna fuente de alimento. Si bien una sola hormiga no podría hacerlo por si misma, es la cooperación de todas ellas para lograr un bien común lo que hace la diferencia en la búsqueda. El objetivo de un algoritmo de hormigas es la solución de problemas de optimización a partir de la simulación de las técnicas utilizadas por dichos animales en la solución de problemas de su entorno.

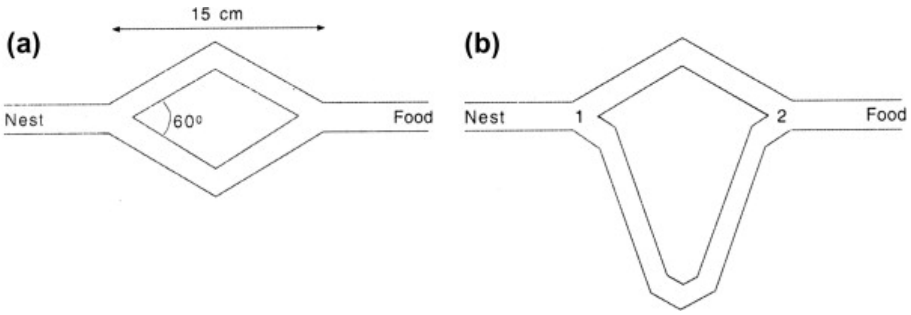

Figura 1.11: Experimento de doble puente con proporción 1:1 y 1:2 respectivamente.

Algunos comportamientos que han motivado a los algoritmos de colonia de hormigas son: exploración, división de tareas y transporte colectivo. Estás actividades las realizan mediante la **estigmergia**, que es un concepto introducido por el biólogo Pierre-Paul Grasse en el año de 1959 y habla sobre una forma de comunicación mediante la modificación del ambiente. Como resultado de esa definición podemos decir que la idea detrás de un algoritmo de hormigas es el uso de una estigmergia artificial para coordinar agentes artificiales [13].

1.3.1 Origen

Existen algunas especies de hormigas que, para encontrar el camino hacia su alimento, utilizan una forma de estigmergia basada en un rastro de feromona en el suelo de modo que otras hormigas al detectarlo, puedan seguirlo [21]. Existen varias especies de hormigas que utilizan un tipo de feromona para dejar marcas en el suelo al momento de buscar el camino hacía una fuente de alimento o hacia el nido. Otras hormigas pueden encontrar esta feromona y decidir si seguir ese camino o no, dependiendo de que tan fuerte es la concentración.

Esta conducta ha sido objeto de estudios e investigaciones por parte de la comunidad científica. Uno de estos estudios es el llamado Experimento de doble puente, realizado por Deneubourg y sus colegas [21]. El experimento consiste en conectar un hormiguero con una fuente de alimento mediante un puente doble y variar la proporción que existe entre los brazos del puente, como se puede ver en la Figura 1.11.

En el primer caso, a pesar de haber demasiadas variaciones al principio, eventualmente todas las hormigas optaron por una rama. En el segundo caso, la mayor parte de las veces escogieron el camino más corto. Esto se debe a que, inicialmente, ambos caminos son idénticos para cada hormiga, sin embargo las hormigas que escogieron el tramo más corto regresan primero al nido y al tener que tomar una decisión sobre que camino escoger, la mayor carga de feromona en el tramo más corto tendrá una mayor influencia en la decisión de las hormigas. De la misma forma, la evaporación de la fe-

romona será menor en el tramo corto debido a que es más probable que las hormigas lo escojan, reforzando así la cantidad de feromona. Este proceso se repite hasta que eventualmente, todas las hormigas convergen a una sola solución.

1.3.2 Hormigas aplicadas a problemas de ruteo

La **optimización por colonia de hormigas** (ACO, por sus siglas en inglés) es una metaheurística de optimización combinatoria que consta de agentes artificiales basados en hormigas que al simular su comportamiento cooperativo, permiten resolver una gran variedad de problemas de teoría de grafos [14]. La idea básica consiste en que por cada iteración, cada hormiga construya una propuesta de solución y deje un rastro de feromona inversamente proporcional al costo que conlleva dicha solución; la feromona incrementará la probabilidad de que otras hormigas sigan esa misma ruta y en cada iteración la feromona total de la instancia disminuirá en un factor constante, de modo que las soluciones más cortas tengan mayor concentración de feromona. La solución aceptada será aquella en la que el costo sea menor. Esta técnica fue desarrollada por Marco Dorigo en su tesis doctoral en 1992 [13] y su desarrollo comenzó al observar los resultados de los experimentos de doble puente sobre hormigas [22] mencionados anteriormente.

Al modelar este comportamiento por medio de la computadora, se hicieron algunos ajustes y se construyó la primer variante de ACO llamada *Ant System* (AS), orientada a obtener rutas mínimas y a atacar el problema del agente viajero. El algoritmo aplicado a un grafo es bastante sencillo (Algoritmo 1), el primer paso es inicializar la feromona τ_{ij} de cada arco o arista, por lo general se utiliza el valor de 1 aunque depende totalmente del problema. Después, cada hormiga se pone en modo búsqueda o *forward* lo que significa que va a explorar el espacio de búsqueda hasta encontrar una solución. Una vez que se construyó una solución se eliminan los posibles ciclos encontrados, se cambia el modo búsqueda de la hormiga y se toma su camino de regreso, depositando feromona en cada adyacencia que conforma la propuesta de solución encontrada. Las acciones locales o daemons son opcionales y la mayoría de las veces se aplican otros procedimientos como búsqueda local para intentar mejorar aún más las soluciones generadas por ACO.

Sea k una hormiga, la construcción de la solución consiste en que k parte del vértice origen v_o y, mientras no haya llegado al vértice destino v_f, recorre por los distintos vértices para construir una ruta. Si k se encuentra en el vértice i, la probabilidad p_{ij}^k (probabilidad de que k avance al vértice j) está dada por:

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^\alpha}{\sum_{l \in N_i^k} \tau_{il}^\alpha} & \text{si } j \in N_i^k \\ 0 & \text{si } j \notin N_i^k \end{cases}$$

Donde N es la vecindad del vértice i, es decir, todos los vértices que comparten una

adyacencia con i sin incluir al vértice anterior en el recorrido de k, a menos que no tenga más vecinos en cuyo caso, debemos regresar sobre nuestro recorrido para evitar estancarnos. Esta construcción puede generar ciclos en el camino por lo que es necesario eliminarlos para conservar el recorrido necesario de la hormiga k para evitar que se deposite feromona de manera inútil. Cada que una hormiga se mueve de un vértice a otro se aplica la fórmula de actualización de feromona $\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \quad \forall (i,j) \in A$ para simular la evaporación de feromona. Cuando la hormiga k logra construir una solución debe actualizar la feromona de la siguiente manera:
$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^k$
Donde $\Delta\tau^k$ es la feromona que la hormiga k deposita.

Algoritmo 1 S-ACO

```
1: procedure S-ACO
2:     while no se alcance la condición de paro do
3:         Construir soluciones
4:         Acciones locales o daemons
5:         Actualizar feromona
```

1.4 Problemas de ruteo

Los **problemas de ruteo** son aquellos que conciernen tanto la distribución de bienes como la de servicios entre depósitos y usuarios finales llamados clientes, los cuales se encuentran en una red de caminos en la que dichos servicios serán brindados por una flota de vehículos en un periodo de tiempo específico [50]. En los últimos años han cobrado relevancia debido a su aplicación en distintas áreas y al impacto positivo que han tenido, sobre todo en logística. Algunos ejemplos de situaciones que pueden ser modeladas como un problema de ruteo son: limpieza de calles, búsqueda de rutas para transporte público, repartidor de combustible, entre otros.

Los problemas de ruteo constan de cuatro partes esenciales para su definición: los clientes, los vehículos, la función objetivo y la red sobre la cual se desarrolla el problema [46].

1.4.1 Red

La **red** representa todas las posibles rutas a seguir dependiendo de las restricciones en distancia, dirección y conexidad entre los distintos componentes del problema de ruteo. La forma más común es mediante un grafo. Se puede hacer una clasificación de problemas de ruteo según el tipo de grafo sobre el cual están representados, de modo que tenemos problemas de ruteo dirigidos, no dirigidos y mixtos. También en la red se ubican los depósitos,que son los lugares de donde parten los vehículos para iniciar su

recorrido, se realiza la carga/descarga de la capacidad del vehículo y puede ser también el punto de regreso de estos.

1.4.2 Cliente

Los **clientes** son los agentes que demandan su producto/servicio y tienen las siguientes características:

- En general se encuentran en los vértices del grafo que representa la red de caminos, aunque pueden encontrarse en los arcos y aristas.
- Tienen una demanda q_i, la cual, debe ser recogida o satisfecha por el vehículo que los visite.
- Pueden tener una ventana de tiempo asociada $W = [w_0, w_1]$, que indica en que intervalos de tiempo pueden atender al vehículo.
- Pueden tener un valor asociado t_i que indica el tiempo que toman en satisfacer su demanda/servicio.
- Si T representa al conjunto de vehículos del problema, un subconjunto $T_p \subseteq T$ de modo que solo los vehículos que pertenezcan a él puedan atender sus peticiones.

A pesar de estar definidos de esta manera, a veces no es posible atender a todos los clientes por lo que, en muchos casos, también se les asigna prioridades o penalizaciones de acuerdo al servicio que solicitan.

1.4.3 Vehículo

Los **vehículos** son los encargados de llevar los bienes del depósito hacia los clientes a través de los posibles caminos en la red y tienen las siguientes características:

- Un depósito origen d_0 y pueden o no terminar su recorrido en ese depósito.
- Un valor Q_i que indica la capacidad del vehículo i, este valor no debe ser sobrepasado en ningún momento.
- Sea $G = (V, E)$ el grafo asociado a un problema de ruteo, $E_v i \subset E$ el subconjunto de arcos asociado al vehículo i, que indica los arcos y/o aristas que puede recorrer.
- Un valor C_i que indica el costo de usar el vehículo i.

1.4.4 Función objetivo

La **función objetivo** modela algunas restricciones del problema de manera matemática buscando el valor mínimo o máximo posible al evaluar alguna solución candidata. La más utilizada es la minimización de distancia, aunque también se puede trabajar sobre el número de vehículos, el tiempo de servicio, el gasto de combustible, entre muchas otras (así como sus combinaciones). Las soluciones van a depender de la función objetivo que se decida usar y del número de variables a las que se aplica.

1.4.5 Problemas de ruteo clásicos

En la literatura encontramos principalmente tres problemas de ruteo, los cuales han sido objeto de estudio por muchos científicos. Los problemas son los siguientes: El problema del agente viajero (TSP, por sus siglas en inglés), el problema de ruteo de vehículos (VRP, por sus siglas en inglés) y el problema del cartero chino (CPP, por sus siglas en inglés). Estos problema son básicos y se les han aplicado bastantes extensiones para adecuarlos a las necesidades actuales desarrollando así, nuevas técnicas que ayuden en la búsqueda de sus soluciones. A continuación se dará una breve descripción de cada uno.

Problema del agente viajero

El TSP o problema del agente viajero se plantea de la siguiente forma: un agente de viajes cuenta con una lista de ciudades y su objetivo es visitar todas las ciudades una sola vez, comenzando y terminando en su ciudad de origen y asegurando que la ruta que decide seguir es la de menor distancia. De manera formal consiste en encontrar un circuito Hamiltoniano de peso mínimo sobre una gráfica $G = (V,E)$ [7], y es quizá, el problema más estudiado de optimización combinatoria.

El origen del TSP es incierto, de acuerdo a Miller-Merbach [37], se menciona el problema en 1831 por [55]. Sin embargo, el primer uso de la expresión *Travelling Salesman Problem* fue hecha por Karl Menger [27]. El problema del TSP, aunque sencillo de plantear, requiere de alrededor de $n!$ operaciones para encontrar la solución óptima, donde n es el número de vértices en G.

Problema de Ruteo de vehículos

El problema de ruteo de vehículos surge a finales de la década de los 50 y consiste en una extensión del TSP en la cual existen varios agentes que tienen una capacidad limitada C y deben visitar a un conjunto de clientes con una demanda q_i de modo que

solo los visiten una vez, no excedan su capacidad y terminen su recorrido en el lugar donde lo iniciaron [10]. Esto fue aplicado a minimizar la distancia recorrida al realizar entregas de gasolina.

El VRP (por sus siglas en inglés) puede definirse de manera formal de la siguiente manera.
Sea $G = \{V, E\}$ un grafo, $V = \{1, 2, \ldots, n\}$. El vértice 1 es el depósito para una flota de vehículos idénticos de capacidad Q que procesan una demanda q_i en cada vértice-cliente $i \in V - \{1\}$. Se asocia un costo no negativo c_{ij} para cada arco $(i, j) \in A, \quad i \neq j$ [42]. El problema consiste en determinar un conjunto de rutas de costo mínimo tal que:

- Cada vértice (además del depósito) debe ser visitado una vez por un único vehículo.
- Todas las rutas de los vehículos empiezan y terminan en el depósito.
- La demanda total en cada ruta no debe exceder la capacidad del vehículo.

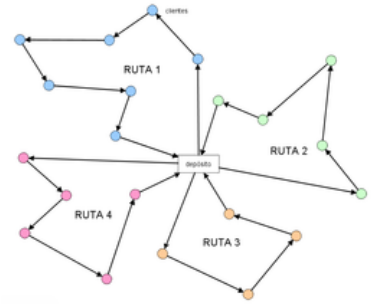

Figura 1.12: Ejemplo de VRP.

Como ejemplo de una situación que se pueda modelar mediante el ruteo por vértices podemos imaginar a una empresa con varios camiones, los cuales deben llevar sus productos a varias tiendas ubicadas en distintas partes de una ciudad. Este problema exige encontrar la mejor ruta para los camiones con respecto a distancia, con la restricción de que los camiones deben salir y regresar al depósito de la empresa; en este caso los vértices del grafo son las tiendas y el depósito, mientras que las aristas conforman las distintas calles por las que pueden transitar los camiones. Esta versión del problema es muy sencilla y se le pueden aplicar muchas más restricciones, pero para fines ilustrativos es suficiente dejarla como está. Además se puede ver claramente que al representar dicho problema como un grafo, el problema gira en torno a los vértices.

Problema del cartero chino y el ruteo por arcos

Hasta ahora los problemas mencionados han trabajado sobre vértices, pero existen también problemas cuyas restricciones obliguen a cubrir los arcos o aristas de un grafo; estos problemas reciben el nombre de problemas de *ruteo por arcos* (ARP, por sus siglas en inglés). El estudio de este tipo de problemas se remonta a los orígenes de la teoría de grafos en el siglo XVIII, cuando Leonhard Euler estudió el famoso problema de los puentes de Königsberg [Figura 1.13]. Este problema consistía en encontrar una ruta que comenzara en cierta parte del mapa, recorriera todos los puentes una vez sin repetirlos y regresara al punto de inicio, es decir, un ciclo Euleriano. Euler demostró que dicho recorrido era imposible y no solo eso sino que, además, su análisis y demostración del problema se convertirían en la base de la formalización de la teoría de grafos.

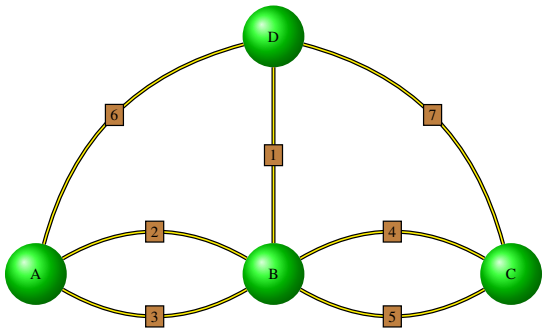

Figura 1.13: El problema de los puentes de Königsberg en un grafo

El revuelo por este tipo de problemas volvió en 1962 con el estudio del problema del cartero chino (CPP, por sus siglas en inglés) por el matemático Mei-Ko Kwan [29], el cual, consiste en encontrar la ruta que minimice la distancia recorrida por un cartero que tiene que entregar correspondencia por todas las calles del pueblo y que, al terminar, regrese a la oficina de correos. A diferencia de los anteriores problemas de ruteo, el CPP puede resolverse en tiempo polinomial si el grafo sobre el que se plantea cumple algunas restricciones [33], en otro caso no existe algoritmo que lo pueda resolver en tiempo razonable.

Tenemos problemas en la vida diaria que, al transformarlos como una instancia de ruteo por vértices, resultan intratables o alteran la esencia original del problema. Un claro ejemplo sería el problema de encontrar la mejor ruta que debe seguir un camión recolector para recoger la basura de alguna colonia; los vértices serían las intersecciones entre calles, las aristas serían las calles en las que el camión debe realizar su trabajo y otras restricciones podrían ser: priorizar el servicio en un subconjunto de aristas o minimizar el número de aristas repetidas y necesitamos asegurar que el camión pase por

todas las calles de la colonia, es decir, que no deje ninguna arista del grafo sin recorrer. ¿Cómo podemos representar esto en un problema de vértices? La respuesta es que no lo podemos hacer, de modo que, necesitamos otro modelo que nos permita plantear este problema de manera adecuada, por lo que definiremos algunas variantes del problema de ruteo por arcos.

1.4.6 Variantes de ruteo por arcos

Al analizar los problemas reales y tratar de plantearlos como un ARP, nos damos cuenta de que entre más complejo sea el problema a resolver, más difícil es encontrar un modelo que nos permita representarlo con exactitud. Debido a esto, han surgido muchas variantes que parten del problema del cartero chino, por esto, es importante hacer la representación matemática formal del CPP.

De acuerdo a Masoud Rabbani y Setare Mohammadi [45], si tenemos un grafo $G = (V,E)$ con $n = |V|$, el problema se puede plantear como un conjunto de ecuaciones matemáticas de la siguiente manera: Si C_{ij} nos indica el costo de ir del vértice i al vértice j y X_{ij} es el número de veces que vamos del vértice i al vértice j lo que buscamos es minimizar las veces que pasamos por cada arista para que el costo sea mínimo, o sea mín $\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij}$.

También debemos de asegurar que cada vez que entramos a cualquier vértice, debemos salir de el, por lo que la ecuación $\sum_{j=1}^{n} X_{ij} - \sum_{j=1}^{n} X_{ji} = 0, \quad \forall i \in V$ es necesaria. Para asegurar que cada arista sea recorrida al menos una vez, necesitamos que la desigualdad $X_{ij} + X_{ji} \geq 1 \quad \forall (i,j) \in E$ se cumpla. Por último, queremos que el número de veces que se recorre cada arista sea un valor entero, esto es, $X_{ij} \geq 0$, $\forall i, j \in V$, $X_{ij} \in \mathbb{Z}$.

Este problema puede ser resuelto en tiempo polinomial [16], lo que significa que existe un algoritmo que nos puede brindar una solución óptima en un tiempo razonable. Sin embargo existen otras variantes que no pueden resolverse en tiempo polinomial [32], pero que vale la pena enunciar ya que al entender éstas, el análisis de algún otro tipo de problema de ruteo por arcos será más sencillo. Con respecto a las variantes del ARP, existen tantas como combinaciones de restricciones se puedan generar, sin embargo, con entender estas variantes básicas podemos adentrarnos más en la teoría de los problemas de ruteo y comprender la mayor parte de los problemas planteados como tal. Corberán y Prins [8] presentaron las siguientes variantes.

- Problema mixto del cartero chino (MCPP, por sus siglas en inglés)
  Dado un grafo mixto $G = \{V,E\}$ con pesos no negativos, encontrar una caminata cerrada de costo mínimo tal que pase por todas las aristas de G al menos una vez.

- Problema ventoso del cartero (WPP, por sus siglas en inglés).
  Similar al MCPP excepto que G es dirigida, por lo que ir de un vértice A a un

vértice B puede tener un costo distinto que el recorrido de B hacia A. Como se puede apreciar, es una generalización del MCPP.

- Problema del cartero chino jerarquizado (HCPP, por sus siglas en inglés).
  Variante del CPP en la que se establece una relación de precedencia entre los diferentes subconjuntos en los que se divide el conjunto original de aristas. Dicha relación indica que si tenemos un subconjunto E_i que precede a un conjunto E_j, entonces las aristas de E_i deben recorrerse antes que las de E_j. En pocas palabras, se debe respetar el orden en el que las aristas deben ser recorridas de acuerdo a su prioridad. Puede resolverse de manera óptima en un tiempo razonable en casos muy especiales.

- Problema de máximo beneficio del cartero chino (MBCPP, por sus siglas en inglés).
  Se establece un valor de beneficio para cada arista $e \in E$ siendo e_i el beneficio obtenido al pasar por la arista e la $i-$ésima vez. El objetivo es maximizar la suma de los beneficios al salir de un vértice v, recorrer todas las aristas y volver a v.

- Problema de ruteo por arcos con capacidad y no dirigido (UCARP, por sus siglas en inglés).
  Sea un grafo $G = \{V, E\}$ no dirigido, y sea $v_i \in V$ un vértice que funciona como depósito con k vehículos idénticos los cuales tienen capacidad Q, además se asocia un costo c_e a cada arista e, $\forall e \in E$. Por último se obtiene un subconjunto de aristas $E_R \subseteq E$ a las cuales se debe visitar. Cada arista tiene una demanda $q_e \geq 0$ y un costo de procesamiento p_e. El objetivo es encontrar un conjunto de rutas para los k vehículos que minimicen el costo de servir a todas las aristas que pertenecen a E_R, de tal manera que cada ruta contenga el depósito y un conjunto de aristas que fueron recorridas por el vehículo de manera que la suma de la demanda de dichas aristas no exceda Q [16].

- Problema de ruteo por arcos dirigidos con capacidad (DCARP, por sus siglas en inglés).
  Similar al UCARP pero en un grafo dirigido [16].

Dentro de los problemas CARP, en el caso de que todas las aristas requieran servicio, es decir, $E_R = E$ el problema recibe el nombre de problema del cartero chino con capacidad (CCPP, por sus siglas en inglés). Este problema también pertenece a la clase de los problemas NP-Duros [2]. Otra cosa importante es que el ARP se puede transformar a una instancia del VRP y viceversa, haciéndolos equivalentes [59]. El método para transformar de VRP a CARP [18] consiste, a grandes rasgos, en construir un TSP generalizado, luego un TSP de *clusters* y después un TSP normal, el cual, puede ser resuelto mediante programación lineal.

Capítulo 2

Estado del arte

Como ya se mencionó en el Capítulo 1, el problema CARP consiste en satisfacer las demandas sobre los arcos de un grafo mediante uno o varios vehículos con capacidad limitada, los cuales inician su recorrido en un depósito y finalizan en el mismo. Además, queremos que el recorrido que realicen los vehículos sea mínimo, es decir, que de todas las opciones posibles, la ruta escogida por el vehículo tenga la menor distancia, tiempo, o cualquier otra variable considerada en la función objetivo.

El problema CARP [20] surge como una extensión al ya conocido problema del cartero rural (RPP por sus siglas en inglés), ya que en este problema no existen restricciones acerca de la capacidad del cartero, o sea, puede realizar su trabajo en un solo viaje sin cargar o descargar su contenedor. Por otro lado, tenemos la problemática de la recolección de residuos en las grandes ciudades, la cual fue una motivación para este trabajo. En épocas recientes, las autoridades han demostrado su interés por optimizar las tareas de recolección debido al problema que representan, y por esto se han ido dejando atrás las rutas obtenidas a través de la experiencia de los operadores y se ha optado por utilizar técnicas que si bien no pueden asegurar soluciones óptimas, ayudan a mejorar en muchos casos las condiciones del trabajo al que se aplican, logrando no solo disminución en tiempo o distancia, sino también una reducción en costos, que para ciudades con gran crecimiento y urbanización también empieza a ser un aspecto importante en la toma de decisiones.

El problema CARP simula de una muy buena manera al problema de recolección en las ciudades ya que se tiene una flota de vehículos, los cuales tienen una capacidad limitada y a su vez deben brindar su servicio en los arcos de un grafo, lo que equivale a las calles de la ciudad. A pesar de que no todos los trabajos mencionados trabajan directamente con el CARP, sus resultados y las técnicas que utilizan demuestran las distintas formas que existen de atacar a los problemas de ruteo, además de que mejoran continuamente colocándose a la vanguardia de la resolución de este tipo de problemas.

2.1 El problema ARP, breve historia

A modo de recopilación, Eiselt. Gendreau y Laporte [16] presentaron un pequeño conjunto de aplicaciones basadas en el RPP como barrido de calles, retiro de nieve en las calles, recolección de residuos, entrega de correo, rutas de transporte escolar y lectura de medidores que se han atacado desde finales de los 70's hasta nuestros días. Como podemos darnos cuenta, todos estos problemas tienen algo en común, que el servicio que ofrecen debe ser a lo largo de una calle y no solo en puntos específicos.
De acuerdo con Wohlk [56] en la década de 1980 las técnicas para resolver el CARP eran heurísticas definidas según las características del problema como escaneo de rutas (*path scanning*) [19] o construir-golpear (*construct-strike*) [15]. Ya en la década de 1990 se comienzan a utilizar metaheurísticas como recocido simulado o búsqueda tabú.

2.1.1 Avances relevantes en la solución de ARP

En el año 2004 se publica el artículo "*First Competitive Ant Colony Scheme for the CARP*", [31] el cual, presenta una aplicación de una colonia de hormigas al problema CARP. La particularidad de esta propuesta es que utiliza dos tipos de hormigas, las elitistas que se encargan de que la solución converja a un mínimo y las no-elitistas que se encargan de explotar el espacio de búsqueda para evitar caer en mínimos locales. A pesar de que los resultados que publicaron fueron buenos, el inconveniente que se encontró fue que se requería de un mayor tiempo de cómputo para resolverlo. Sin embargo, posiciona a ACO como una metaheurística competitiva en el ámbito del ruteo.

En el artículo "*Solving Capacitated Arc Routing Problems using a transformation to the CVRP*" [35] se propone una técnica de solución para el CARP en la que cada arco se reemplaza por dos vértices para convertir el problema a una instancia de CVRP. La restricción es que los vértices que forman un arco deben ser visitados de manera consecutiva para simular el recorrido por el arco original. Para resolver este VRP se utilizó la técnica de *Branch and Cut and Price* [17]. Este trabajo es trascendente porque se resolvieron por primera vez todas las instancias del conjunto de *benchmark* gdb de manera óptima, además de que se encontraron tres nuevos óptimos, uno para el conjunto de *benchmark* val y dos para egl.

Para el problema de ECARP [30] el cual extiende al CARP con restricciones más realistas como la prohibición de ciertas vueltas o la aplicación de un servicio en ambos lados de la calle, se desarrolló en el año 2011 un algoritmo de hormigas híbrido llamado HACOA [57], el cual es una colonia de hormigas que incorpora técnicas de MMAS y además utiliza información heurística dependiente del estado del grafo, así como un ajuste dinámico de parámetros y optimización local. Esta técnica demostró superioridad a MMAS, como era de esperarse, y demuestra que sí vale la pena añadir nuevas técnicas que ayuden a la colonia de hormigas a mejorar la calidad de sus soluciones.

En el artículo llamado "*Node, Edge, Arc Routing and Turn Penalties: Multiple Problems—One Neighborhood Extension*" [53], se probaron 1528 instancias de 18 distintos *benchmarks* utilizando la búsqueda local iterada [44] y búsqueda genética híbrida unificada [54] en distintos tipos de problemas de ruteo como el CARP, NEARP, NEARP con restricción en las vueltas, PCARP entre otros. La novedad de su propuesta es que utiliza programación dinámica para obtener información, la cual influirá en cada toma de decisión que realice el algoritmo mientras construye la solución. A pesar de que teóricamente ya se conocía esta estrategia, se tenía la idea de que el tiempo computacional que llevaría hacer todas esas búsquedas heurísticas sería demasiado. Sin embargo, logran realizar esta búsqueda en tiempo constante amortizado.

Al realizar las pruebas se utilizaron los mismos parámetros de las técnicas originales con la adición de la búsqueda de vecindarios, logrando obtener mejores resultados y en un tiempo menor. Esta técnica resulta aún más poderosa ya que fue probada en distintas variantes del ARP e incluso en los conjuntos de *benchmark* más difíciles para las heurísticas hoy en día como: EGL, DI-NEARP y CBMix. Por otro lado, la técnica resultó poco eficiente en problemas transformados a variantes de VRP, por lo que el autor considera que, la explotación de información del vecindario en la construcción de la solución, beneficia más a la solución que la transformación a VRP. Con esto, se espera que futuros trabajos realicen un mayor aprovechamiento de la información que se pueda obtener del grafo.

2.2 Aplicaciones reales a problemas de recolección y similares

A continuación se van a presentar algunos trabajos sobre ruteo y optimización de recolección de residuos sólidos en distintas partes del mundo. La importancia de estos trabajos es que todos son problemas reales y han resuelto o mejorado las soluciones con las que ya contaban. Esto es solo una muestra más de que la aplicación de las técnicas descritas anteriormente funcionan y si bien, no podemos simular completamente un problema real, al menos podemos atacarlo de una manera u otra con la certeza de obtener alguna mejora.

En el año 2002 se publicó el trabajo "*The Application of a Vehicle Routing Model to a waste-collection problem: two case studies*" [1]. En él, aplicaron algoritmos con búsqueda tabú en casos de estudio en la región de Val Trompia en Brescia, Italia y el área urbana de Antwerp en Bélgica. En el primer caso buscaban evaluar el desempeño de los carros recolectores de contenedor fijo contra los de contenedor removible. En el segundo caso se evaluaron distintos modelos de recolección en el mismo tipo de carros recolectores. La motivación de estos autores fue que el incremento de desechos en distintas ciudades ha ocasionado que el manejo de los mismos sea considerado como un problema, además de que es difícil elegir un método para disponer de la basura, ya

que hay una gran variedad y cada uno tiene ventajas y desventajas que afectan tanto a la población como al gobierno.

En el artículo, Angelelli y Speranza manejan tres sistemas de recolección: tradicional, carga lateral y carga lateral desmontable. El método tradicional consiste en que el camión pase frente a cada casa y los trabajadores que van en el camión se encargan de depositar las bolsas en los camiones. Este sistema requiere de mayor personal por cada camión y de un esquema de recolección constante. El segundo sistema consiste en que la personas lleven su basura a depósitos especiales para que cuando el camión llegue a dicho depósito, se coloque a lado de el y mediante un mecanismo semi-automático, coloque los desechos dentro del camión. En general solo se necesita una persona para operar el mecanismo, por lo que requiere menos personal que el primer método. Además es más rápido que el primer sistema ya que no requiere tanto tiempo para recoger todos los desechos y no tiene que pasar por todas las calles sino solo por donde están los depósitos. Sin embargo, para las personas esto resulta ser menos cómodo y requiere de su participación para que esto sea eficiente. El tercer sistema es similar al segundo excepto que, el camión puede dejar su contenedor cuando esté lleno y colocarse uno vacío. Esto hace que la fase de recolección este separada de la fase de transporte, pero aumenta los costos ya que estos vehículos son mucho más caros que los de contenedor fijo.

Al aplicar los modelos a las situaciones reales lo primero que se hizo fue estimar todas aquellas variables que son no deterministas para poder volver el problema a algo determinista. El primer caso de estudio consistió en una instancia de 181 vértices mientras el segundo tuvo un tamaño de 345 vértices. El modelado se realizó con un sistema llamado macropuntos el cual consiste en agrupar vértices cercanos con las mismas características (tipo de desecho, día de recolección, cercanía, etcétera) en un solo punto. Cada macropunto contiene la lista de puntos reales que lo conforman y un conjunto de posibles calendarizaciones (las cuales cumplan con tiempos y capacidad) para las visitas. La distancia total dentro de un macropunto se distribuye de manera uniforme entre sus miembros y de esta manera se puede trazar la gráfica del problema, pues ya se sabe la distancia entre puntos de distintos macropuntos.

Para la solución se utilizó el siguiente algoritmo: Se escoge una calendarización de visita para cada macropunto de manera aleatoria; a todos los contenedores dentro del macropunto se les asigna este esquema para lograr que el conjunto de contenedores visitados cada día sea constante. El algoritmo se comienza a "mover" de una solución a otra mediante pequeños ajustes a la solución actual. Después de un cierto número de iteraciones, se devuelve la mejor solución. Los movimientos válidos por el algoritmo son:

- Pasar un contenedor de una ruta a otra en el mismo día.
- Asignar un esquema distinto a un contenedor.
- Dos rutas son interrumpidas y se cambian sus adyacencias.

- Se redistribuyen los contenedores de dos rutas.

Una vez que se generaron todas las soluciones, se toma la mejor (aquella que minimiza una función de costo) que no esté en la lista tabú. La función de costo evalúa la función objetivo y las penalizaciones por violación a las restricciones de tiempo y capacidad. La lista tabú lleva un registro de movimientos prohibidos de modo que no se haga el movimiento contrario en iteraciones siguientes, pero ésta tiene un límite de $O(\log n)$, donde n es el número de contenedores.

En el caso de Val Trompia, el método de recolección es el de carga lateral y la empresa que se encarga de brindar el servicio quiere ver si es conveniente mantener ese sistema o pasar al sistema de carga lateral desmontable. Como el cambio no afecta a los habitantes de la región, el factor económico fue crucial para la decisión. Después de optimizar el problema, se llegó a la conclusión de que el método utilizado hasta ese momento resultó ser menos rentable que el nuevo método propuesto al utilizar un camión con caja desmontable y un tráiler para transportar los contenedores llenos de residuos al incinerador.

En 2012 se publicó un artículo llamado "*Ant Colony Algorithm for Waste Collection Vehicle Arc Routing Problem with Turn Constraints*" [34]. En él, los autores aplicaron la técnica ACO para optimizar las rutas de recolección en la ciudad de Shuangnan. Entre las restricciones que impusieron al modelado del problema, se incluyó la prohibición de algunas vueltas debido al ángulo tan estrecho que formaban. Para resolver el problema, se transformó la instancia del problema a un VRP y se aplicó la técnica de ACO específica para núcleos. Al aplicar la optimización sin restricción sobre las vueltas se redujo la distancia a recorrer en 31.1 % y al aplicar la restricción se redujo la distancia en 31.9 %.

Para realizar la transformación a VRP se realizó el siguiente procedimiento: Sea $G' = \{V, E\}$ el nuevo grafo, cada vértice $v \in V$ ayuda a representar una adyacencia del grafo original utilizando un solo vértice en caso de querer representar un arco y dos vértices si se desea representar una arista, mientras que el vértice 0 representa el depósito. Debido a lo anterior, el conjunto $V = T_o \cap t_1 \cap \ldots \cap T_r$ donde cada T_k es la representación de una adyacencia en el grafo original, cabe destacar que cada T_k tiene a lo más dos vértices. La arista (P_{ij}, Q_{kh}) existe si se encuentra un camino entre i y h en el grafo original que cumpla las restricciones del problema, el costo de la arista es su camino de distancia mínima. Si existe una vuelta prohibida se busca el camino más congruente. Después se aplica un sistema de colonia de hormigas basado en *clusters*, se usan tantas colonias de hormigas como vehículos, obteniendo rutas para cada vehículo. Cuando se llena el vehículo, este se desplaza hacia el depósito para liberar su carga y después continua hasta que todos los vértices hayan sido visitados o termine el tiempo. La actualización de feromona será afectada por la mejor ruta global y la mejor de la ejecución actual, evitando quedar atrapado en un mínimo local.

Las pruebas se realizaron con y sin restricción de vueltas, cada caso fue probado

10 veces con los siguientes valores:
$\alpha = 1,\ \beta = \gamma = 5,\ \rho = \varepsilon = 0.4,\ q_0 = 0.9,\ s = 10$ y 500 iteraciones. Los resultados mostraron un decremento en la distancia de las rutas de 31.1 % y 31.9 % con y sin restricción de vueltas respectivamente. Además, descubrieron que bastan seis vehículos para poder realizar las tareas de recolección, lo cual, reduce costos de operación. En "Using Ant Colony Optimization to solve Periodic Arc Routing Problem with Refill Points" [49], los autores utilizaron la técnica ACO para resolver una instancia de ARP periódico con puntos de recarga (PARPRP). Esta instancia era la transformación del problema de regado de árboles con un nodo cisterna en la ciudad de Kaohsiung. Se eligió el ARP periódico ya que los árboles tenían distintas necesidades de riego y cada una de ellas es un ARP. Además, se utilizó una estrategia de búsqueda local y se hizo una transformación del grafo para poder resolver el problema como una instancia del TSP.

Para resolver este problema se realizó la transformación de Lamport la cual consiste en asignar un vértice a cada arco y arista del grafo. Se crean los "hiper-vértices" $X =< u,v >$ tal que u representa el vértice de inicio y v el vértice de fin en el grafo original. El origen y los depósitos se quedan como $< o,o >$ y $< d,d >$. Después de haber creado los hiper-vértices, se crearon los hiper-arcos de la siguiente manera:
Un hiperarco se define como $R < Xu,Yv >$ donde X,Y son hipernodos, u representa el vértice de salida y v el de entrada, de tal manera que dos hipernodos con un nodo en común son adyacentes. Si tenemos r arcos en G, tendremos r^2 hiperarcos en G'.

Una vez que se define lo anterior se puede aplicar ACO a este nuevo grafo para resolver el PARPRP. Como el problema se debe resolver para varios días, entonces se tiene el mismo grafo varias veces pero con distintos clientes a satisfacer, en este caso, los árboles. El número de vehículos se inicia liza en una cierta cantidad y se decrece hasta el punto de ya no poder obtener soluciones posibles, en ese momento se dice que el mínimo número de vehículos requeridos ya fue alcanzado. Éste es el primer objetivo de ACO.

Una vez obtenido el primer objetivo, se busca minimizar la distancia utilizando ACO de manera usual, mezclando además un algoritmo de búsqueda local que consiste en cambiar dos nodos en la solución de manera que se visiten en orden diferente y ver si la solución mejora. Estos dos nodos se eligen de manera que no violen las restricciones del problema, o sea, que sean visitados por el mismo vehículo el mismo día. Se realizó la corrida de la simulación con 1, 2, 3, 4 y 5 puntos de relleno de agua para los camiones y en todas se observó una mejora de la ruta que se tenía con respecto a la nueva ruta.

En el artículo llamado "*Capacitated Clustering and Collection of Solid Waste in Kwadaso Estate, Kumasi*" [39] trabajaron con la zona de Kwadaso en Ghana para resolver su problema de recolección. Dicha área contaba con aproximadamente 590 contenedores, cada uno con 240 litros de capacidad. Se dividió el área en núcleos de acuerdo a la capacidad del vehículo recolector que se le asignaba y a cada vehículo se le asignó toda la recolección de un núcleo en un día. Con esta división, utilizaron ACO para encontrar la ruta mínima a seguir por los vehículos. Lo primero que se hizo fue clasificar a los depósitos en *clusters* de acuerdo a la máxima demanda y mínima

distancia, esto para evitar la creación de clusters adicionales. Los pasos para realizar esto son:

- Calcular k, el número de clusters donde $k = \sum_{i=1}^{n} \frac{d_i}{Q}$, donde Q es la capacidad del clúster.
- Se ordenan los depósitos de acuerdo a su demanda y se escogen los primeros k para ser centroides.
- Se calculan las distancias entre todos los solicitantes y los centroides, para obtener la prioridad y asignarles un centroide.
- Se itera hasta que la clasificación ya no cambia.
- Se ejecuta ACO donde las hormigas tendrán una lista tabú que solo les permitirá hacer movimientos legales.

Después de ejecutar ACO, se logró obtener buenos resultados tanto en distancia como en costos, logrando reducir el número de camiones utilizados de 8 a 6, reduciendo costos de recolección y aumentando la frecuencia del servicio debido al sistema de clústers.

En "*Application of Arc Routing Problem for Municipal Solid Waste Collection in Kolhapur City, India*" [4], los autores plantean el problema de recolección de residuos sólidos en la ciudad de Kolhapur, particularmente el área de Rajarampuri. Su desarrollo es bastante simple ya que después de obtener los puntos de recolección de residuos para la zona, cinco en total, se establecen las distancias que existen entre cada uno de los puntos, esto con la finalidad de obtener una gráfica completa.

Una vez obtenidas las distancias, se puede plantear el problema como un TSP, para el cual inicializan ACO, en la que las hormigas tendrán una lista tabú para marcar los vértices ya visitados. De esta manera convirtieron un pequeño problema de VRP a una instancia del TSP para poder aplicar ACO y obtener la ruta óptima de recolección de residuos ubicados en cada uno de los puntos. En esta aplicación solo se consideró la distancia como una variable real del problema.

En el año de 2015 el trabajo "*Optimal routing for waste collection: a case study in Singapore*" presenta una alternativa de solución al problema de recolección de residuos en Singapur [58]. La solución fue construida mediante el uso de la técnica ACO min-max (MMAS, por sus siglas en inglés), acompañada del algoritmo de Dijkstra para encontrar la mejor ruta a tomar desde los puntos de generación de residuos hasta los incineradores de basura de la ciudad considerando tiempos de traslado, probabilidad de accidentes y exposición a la población con el fin de reforzar la decisión tomada. Después de obtener todos estos datos, el modelo se implementó de manera exitosa en Singapur y se demostró su efectividad.

Para la colonia de hormigas utilizada por Xue y Cao, se establece que el valor de heurística será la distancia euclidiana del vértice actual al objetivo. Las hormigas iniciarán en el origen y tendrán como objetivo el vértice destino. Además, una vez que una hormiga se encuentre atrapada y no pueda realizar más movimientos, se detiene su ejecución y se continua con la ejecución de la siguiente hormiga, incrementando así eficiencia y mejorando la aplicación de feromona en la gráfica. Si todas las hormigas se atoran en una ejecución, se repite este paso. Una vez que las hormigas han construido sus rutas se procede a la evaluación de las mismas. Los criterios para evaluarlas son:

- Tiempo de traslado, se busca minimizar.
- Riesgo de accidente: también se busca minimizar
- Exposición a la población: se deben evitar zonas residenciales o comerciales tanto como sea posible, además esto reduciría el daño en caso de accidente.

Después se construye una función objetivo utilizando los valores mínimos y máximos obtenidos, además de los valores obtenidos con Dijkstra y, por último, se busca una solución óptima de Pareto ya que el objetivo es multicriterio. La ejecución de la colonia de hormigas encontró soluciones que obtenían resultados mejores de manera global a las tres rutas encontradas por el algoritmo de Dijkstra con respecto a tiempo, riesgo y exposición, es decir, dominaban a estas soluciones y se convertían en un óptimo de Pareto. Se realizó este mismo procedimiento pero con pesos asignados a cada criterio, de modo que las soluciones variaron un poco y tuvieron distintas inclinaciones dependiendo de los pesos asignados. Dichas soluciones también fueron consideradas buenas. Sin embargo, no fueron superiores del todo a las obtenidas mediante el primer método.

Otro método utilizado es el VRPSD (VRP con demanda estocástica), el cual se planteó en 2009 en el trabajo titulado "*Ant Colony Optimization for Solving Solid Waste Collection Scheduling Problems*" [25] en la ciudad de Johor Bahru, Malasia, en la que una o varias calles agrupadas se representan mediante un vértice con coordenadas únicas. Cada vértice contiene un rango de demanda de servicio posible donde cada elemento del rango tiene la misma probabilidad de aparecer que cualquier otro; la demanda real solo se sabe hasta que el vehículo llega al vértice. Debido a esto, en cada vértice que un vehículo visite debe decidir si continuar al siguiente o regresar al depósito. Debido a que la colonia de hormigas común tiene problemas en instancias de más de 30 vértices, se aplica un sistema de colonia de hormigas (ACS, por sus siglas en inglés), cuyas diferencias radican en:

- La regla de transición equilibra la exploración y la explotación de la ruta actual
- La actualización global solo se aplica a las aristas de la mejor solución
- Mientras las hormigas construyen su ruta, se aplica una actualización de feromona local.

Así mismo, se realizó la comparación de los resultados de la colonia de hormigas contra una implementación de recocido simulado (SA, por sus siglas en inglés) en instancias que van de los 12 a los 48 vértices. Al graficar los resultados se observa que ACS obtiene mejores resultados que SA, ya que tiene una desviación menor con respecto de la mejor solución por lo que es más consistente y fiable, ya que este comportamiento se observó en todas las instancias. Por otro lado SA empeora su desempeño mientras el número de vértices aumenta.

Capítulo 3

Descripción del problema

Este capítulo tiene como fin dar un panorama general de la situación en la delegación Milpa Alta con respecto al manejo de los residuos sólidos, más específicamente del proceso de recolección. Después describiremos su situación actual y la infraestructura con la que cuenta para realizar dicha actividad.

3.1 Antecedentes

La Ciudad de México cuenta con una extensión territorial de 1,485 km^2 y se divide en 16 delegaciones con un total de 1863 colonias. En ella habitan alrededor de 9 millones de personas y se generan diariamente 12,920 toneladas de basura, lo que significa que cada habitante produce en promedio entre 0.86 y 2.44 kg de desechos. El origen de los residuos generados diariamente proviene principalmente de 3 fuentes: domicilios (48 %), comercios (26 %) y sector de servicios (14 %). Las delegaciones que más residuos generan son Iztapalapa, Gustavo A. Madero y Cuauhtémoc, aportando 41 % del total, por otro lado, las delegaciones que menos residuos generan por día son Magdalena Contreras, Cuajimalpa y Milpa Alta aportando el 4 % del total.

En la Ciudad de México se cuenta con 2652 vehículos recolectores de los cuales 1698 son de carga trasera y de doble compartimiento, 155 rectangulares, 39 tubulares, 291 de volteo y 469 de otros tipos. Para realizar la operación de estos vehículos en las 1828 colonias donde se brinda el servicio, se cuenta con 6507 choferes y 1781 rutas de recolección.

Para el manejo de los residuos sólidos, la Ciudad de México cuenta con 12 estaciones de transferencia, dos plantas de selección, dos plantas compactadoras, ocho plantas

de composta y cinco sitios de disposición final. Cada delegación se encarga de brindar servicios de limpia y recolección a sus colonias. Los camiones hacen un promedio de dos viajes diarios para transportar la basura a las 12 estaciones de transferencia antes mencionadas. De acuerdo al Manual Técnico sobre generación, recolección y transferencia de residuos sólidos municipales [48] los tres métodos más comunes de recolección de residuos son *de parada fija*, *de acera* y *de contenedores*.

El primer método, también llamado método de esquina, consiste en comunicar la llegada del camión a la esquina de la calle por medio de una campana. Al escuchar el aviso de llegada del camión, los usuarios acuden a entregar sus residuos a la esquina correspondiente.

El segundo método consiste en recorrer todas las calles de una región y recoger la basura frente a cada domicilio. La basura debe ser colocada poco antes de la llegada del camión. El camión debe contar con dos personas encargadas de colocar los desechos en el camión y el chofer debe activar el mecanismo de compactación cada que sea necesario.

El tercer método consiste en colocar depósitos en lugares estratégicos y accesibles a los camiones. La basura se coloca en los contenedores sin importar la hora y los camiones solo se trasladan a cada uno de los contenedores. Las principales ventajas y desventajas de cada método se indican en la siguiente tabla. Aún cuando existe tecnología disponible para el diseño y la planeación de las rutas de recolección, el sistema más usado para su diseño sigue siendo la experiencia y juicio de los jefes de limpieza y choferes de los vehículos recolectores. Este criterio no siempre es el mejor por lo que, la mayoría de las rutas son ineficientes, provocando fallas y pérdidas en el sistema de recolección como desperdicio de personal, deficiencia de operación y funcionamiento del equipo y reducción de cobertura del servicio. Es por esto que aplicar técnicas computacionales al problema de generación de rutas resulta necesario.

Tabla 3.1: Ventajas y desventajas de los métodos de recolección

Método	Ventajas	Desventajas
De parada fija	- Es un método económico.	- Los usuarios tienen que trasladarse a la ubicación del camión. - Riesgo de tiraderos clandestinos. - Por pereza o por no coincidir con los horarios del camión, existe el riesgo de que los residuos terminen en la calle.
De acera	- Cómodo para los usuarios ya que no se tienen que trasladar ni cargar la basura por mucho tiempo.	- Debe tener un horario y una frecuencia cumplida. - Los usuarios deben sacar sus desechos en el momento adecuado (ni mucho tiempo antes, ni mucho después). - Si los usuarios sacan sus residuos con mucha anticipación, se corre el riesgo de que los animales rompan y/o rieguen la basura.
De contenedores	- Eficiencia y rapidez. - Se puede depositar basura sin restricción de horario.	- Se requieren contenedores y camiones especiales. - Los usuarios deben trasladarse al contenedor. - El servicio debe ser frecuente, en caso contrario, se generan focos de infección por exceso de basura.

3.2 Datos de la delegación

La delegación Milpa Alta es la segunda demarcación más grande de la Ciudad de México, tiene una extensión territorial de 228 km^2 y se ubica al sureste de la ciudad, como se observa en la Figura 3.1. De acuerdo a datos del INEGI, la delegación Milpa Alta cuenta con una población de 115,895 habitantes siendo la delegación con menos personas en la Ciudad de México [12]. En ella se generan 119 toneladas de basura al día correspondientes a 0.86 kg por habitante y cuenta con 58 vehículos recolectores, de los cuales seis son de carga trasera, nueve de doble compartimiento, 27 de volteo y 16 de otras características. Milpa Alta brinda el servicio de recolección a 12 colonias, a través de 86 rutas distintas [11]. Cabe resaltar que el nivel de eficiencia en la recolección de residuos orgánicos de esta delegación es del 80 %, siendo la mejor delegación de la ciudad en este ámbito. En la Tabla 3.2 se mencionan las 12 colonias de Milpa Alta

Figura 3.1: Ubicación de Milpa Alta en la Ciudad de México

así como su extensión territorial y el número de habitantes. Por otro lado, en el mapa mostrado en la Figura 3.2 se puede ver su ubicación. La colonia que se tomó en cuenta para la realización de este trabajo es Villa Milpa Alta.

Tabla 3.2: Colonias de Milpa Alta

Colonia	Extensión Territorial (has.)	Población
San Pedro Atocpan	87.65	sin datos
San Salvador Cuauhtenco	60.77	12543
Villa Milpa Alta	358.69	17957
San Bartolomé Xicomulco	60.77	4155
San Francisco Tecoxpa	59.44	10030
Santa Ana Tlacotenco	174	9833
San Lorenzo Tlacoyucan	122.63	3796
San Juan Tepenahuac	41.63	sin datos
San Agustín Ohtenco	11.16	sin datos
San Antonio Tecomitl	198.6	21714
San Pablo Oztotepec	127	14030
San Jerónimo Miacatlan	29.33	sin datos

Figura 3.2: Pueblos de Milpa Alta

Capítulo 4

Metodología

En el capítulo anterior se habló del problema de obtención de rutas para los camiones recolectores y además se describió el entorno de la delegación Milpa Alta. En este capítulo se abordará la metodología llevada a cabo para la solución del problema, es decir, los pasos realizados para obtener una solución partiendo del mapa urbano del pueblo elegido para la realización de este trabajo; desde la transformación del mapa a un grafo mixto representado como una matriz de adyacencias, así como la aplicación de la técnica de optimización describiendo los parámetros usados, configuraciones y sus restricciones. Por último se hablará de la construcción de las rutas obtenidas a partir de los resultados que se obtengan al aplicar la optimización. La Figura 4.1 muestra

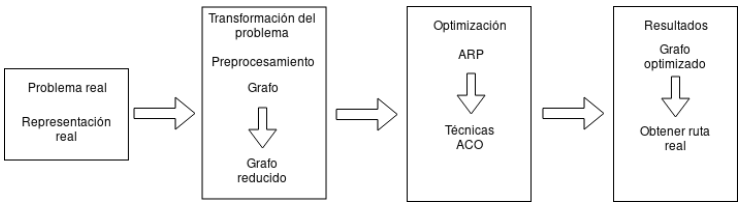

Figura 4.1: Metodología propuesta

los pasos a seguir para resolver el problema, cada uno de los recuadros representa un paso esencial para la construcción de la solución y cada una de ellos está compuesto de tareas más pequeñas y específicas. También se hablará sobre los *benchmarks*, que resultan útiles para la verificación del buen funcionamiento de la técnica de optimización a utilizar. Debido a que los resultados son el objetivo primordial de este trabajo, se les dedicará el siguiente capítulo para su presentación y análisis.

4.1 Transformación del problema

En esta sección se revisará el proceso mediante el cual se transforma el problema real a un grafo. Se detallará cada una de las herramientas utilizadas para la manipulación y conversión del mapa

En esta sección se revisará el proceso mediante el cual se transforma el problema real a un grafo. Esta acción es crucial para la obtención de resultados ya que es necesario la interpretación de los datos reales para la correcta aplicación de la técnica de optimización. Se describirá a detalle cada uno de los pasos realizados así como el uso de cada una de las tecnologías.

4.1.1 Mapa original

Debido a que la delegación no contaba con un mapa actualizado de las calles donde se presta el servicio de recolección de residuos en Milpa Alta, se realizó una búsqueda de plataformas de terceros que permitieran el uso y descarga de sus mapas. Esta búsqueda solo arrojó una opción llamada © OpenStreetMap contributors, que es una página *web* que proporciona mapas de todo el mundo y además permite descargar regiones delimitadas por el usuario. Estos mapas son editados por la comunidad y se permite el libre manejo de los mismos y los datos se pueden usar bajo la licencia de *Open Database License*.
El mapa descargado corresponde al pueblo de Villa Milpa Alta (4.2). La descarga se

Figura 4.2: Vista del mapa a descargar a través de © OpenStreetMap

obtiene en un formato .osm el cual resulta sencillo de manipular, ya que este guarda información relevante como nodos, caminos y distancias. Una vez realizada la descarga,

se puede visualizar el mapa utilizando *software* específico. En nuestro caso, se utilizó *Java Open Street Maps* ya que no solo permite la visualización del mapa urbano, sino que además permite editar la información del mismo. Este *software* resulta crucial para el siguiente paso que consiste en eliminar información no necesaria del mapa para la solución de nuestro problema.

4.1.2 Limpieza del mapa

Para lograr obtener una representación matemática formal del problema, únicamente se tomará en cuenta la información necesaria para construirla, es decir, calles (arcos), intersecciones de calles (nodos o vértices) y las distancias entre cada intersección (pesos de cada arco).

Existen dos principales razones por las que se realizó una limpieza al mapa. La primera se debió al número excesivo de nodos utilizados para dar precisión a las calles, ya que se utilizaban sobre todo para representar curvas; si reducimos la cantidad de vértices en el grafo al mínimo, entonces reducirá la dimensión de la matriz de adyacencias. La otra razón es que existen muchas calles que se utilizaban como privadas, andadores, callejones, entre otras por lo que resultan imposibles de transitar para un camión recolector; es por esto que es innecesario considerarlas para la construcción de las rutas.
En la Figura 4.3 se observa el estado de una calle antes y después de eliminar los nodos

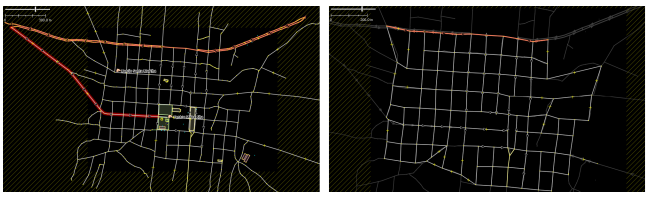

Figura 4.3: Antes (izq.) y después (der.) de la eliminación de nodos de precisión

de precisión. Con esta acción pasamos de tener cerca de 500 nodos a solo 105, siendo una reducción considerable en la magnitud del problema.

En el mapa de la Figura 4.4 se puede observar el resultado final de la limpieza del grafo. Se aprecian mucho menos vértices o nodos, por otro lado, las calles que no son necesarias se pintaron de gris. Este es el mapa definitivo con el que se construirá la matriz de adyacencias.

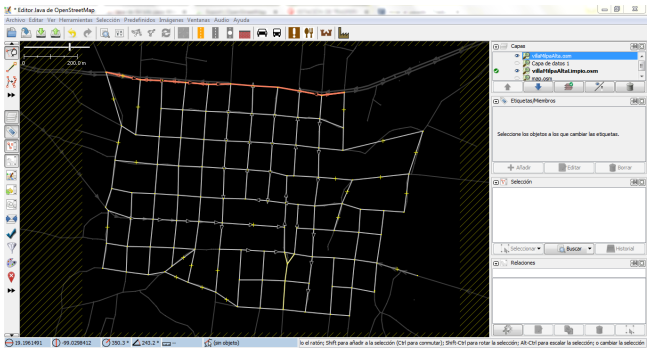

Figura 4.4: Mapa después del proceso de limpieza; las calles sombreadas se eliminaron.

4.1.3 Obtención del grafo

Una vez que se obtiene el mapa definitivo sobre el que se va a trabajar, pasamos a la construcción de la matriz de adyacencias que representará al grafo mixto. Como primer paso, construimos un objeto de tipo JSON (*Javascript Object Notation*, por sus siglas en inglés) utilizando como llaves los identificadores de los nodos; a cada llave le corresponde un objeto que incluye sus adyacencias y su ubicación geográfica. Por ejemplo, el grafo representado en la Figura 4.5 se puede representar como el código 1:

```
{
   "1":{
      "edges":[
         { "node":"2", "distance":1 },
         { "node":"3", "distance":1 }
      ],
      "latitude":0,
      "longitude":0
   },
   "2":{
      "edges":[
         { "node":"4", "distance":1 }
      ],
      "latitude":4,
      "longitude":1
   },
   "3":{
      "edges":[],
      "latitude":8,
      "longitude":0
   },
   "4":{
      "edges":[
         { "node":"1", "distance":1 },
         { "node":"3", "distance":1 }
      ],
      "latitude":4,
      "longitude":-1
```

```
    }
}
```

Código 1: Grafo representado en formato JSON

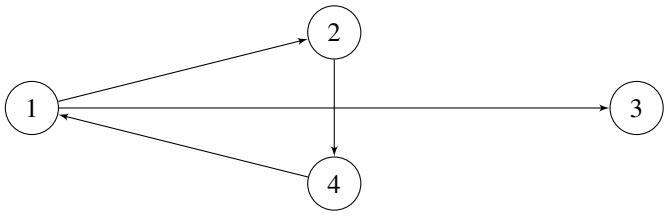

Figura 4.5: Ejemplo de grafo

Cada elemento de la lista de adyacencias incluye un identificador al nodo con el cual comparte un arco y la distancia asociada a dicho arco. A partir de este objeto, se puede iterar fácilmente para construir la matriz de adyacencias, la cual es necesaria para aplicar la técnica de optimización elegida.

4.2 Optimización

Una vez que se realizó la limpieza del mapa y se obtuvo un formato adecuado para trabajar, el siguiente paso es la aplicación de la técnica de optimización a nuestro problema. La técnica que se eligió está basada en la descripción del algoritmo de colonia de hormigas utilizado en el artículo "*An Ant Colony Optimization for the Capacitated Arc Routing Problem*" [51], el cual cuenta con seis etapas que serán descritas en la siguiente sección.

4.2.1 Técnica

Como se mencionó, el algoritmo cuenta con seis etapas. En la primera se inicializan los parámetros y se asigna el valor inicial de feromona a cada arco. El segundo paso consiste en la construcción de una solución válida para nuestras restricciones. Para construir una solución es necesario asignar un vehículo inicial y agregar arcos a su ruta; dichos arcos se seleccionan de acuerdo a una regla de transición. Una vez que ya se visitaron todos los arcos requeridos, finalizamos el proceso de construcción de solución. En caso de que el vehículo exceda su capacidad, dejamos de añadir arcos a la ruta de éste y repetimos el proceso con un nuevo vehículo.

Una vez finalizada la construcción de una solución válida, se aplica la regla de actualización local con el fin de modificar los niveles de feromona (esta regla será descrita

en el apartado de parámetros). En el momento en que todas las hormigas han concluido este proceso, inicia la etapa de búsqueda local. En este cuarto paso, los autores originales utilizan tres técnicas de búsqueda local que son: intercambio, 2-óptimo e inserción. Sin embargo, para este trabajo solo se utilizará la técnica de 2-óptimo. Esta técnica fue propuesta por G. A. Croes [9] en 1958 y consiste en realizar intercambios en el orden en que se visitan los arcos para evaluar el nuevo costo; este proceso se realiza hasta que ya no se obtenga mejora alguna. Después de realizar el proceso de búsqueda local, se aplica la regla de actualización global de feromona que será descrita más adelante. Los pasos anteriores se realizan hasta que se complete el número total de iteraciones.

Para representar una solución se utiliza una lista de arcos los cuales están unidos de manera implícita por las distancias más cortas entre ellos. De este modo resulta sencillo realizar operaciones con las rutas como calcular su distancia total, actualizar feromona, entre otras. Para distinguir las rutas entre un vehículo y otro, se utiliza el símbolo 0. Para ejemplificar esto se muestra la Figura 4.6 que representa una solución de tres vehículos para ocho arcos. El primer vehículo cubrirá en su ruta a los arcos 5, 6 y 8. El segundo a los arcos 1 y 3, y por último, el tercer vehículo cubrirá los arcos 2, 4 y 7.

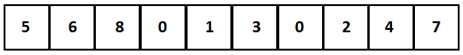

Figura 4.6: Representación de solución

4.2.2 Parámetros

Para la ejecución del algoritmo se utilizarán tres funciones distintas: la función de transición, la actualización local de feromona y la actualización global de feromona. La función de transición tiene como objetivo seleccionar el siguiente arco para continuar la construcción de la solución.

$$s = \begin{cases} arg\,\text{máx}_{j \in N_i}\{\tau_{ij} \cdot \eta_{ij}^{\beta}\} & \text{si } q \leq q_0 \\ S & \text{si } q > q_0 \end{cases}$$

$$S : P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij} \cdot \eta_{ij}^{\beta}}{\sum_{w \in N_i} \tau_{iw} \cdot \eta_{iw}^{\beta}} & \text{si } j \in N_i \\ 0 & \text{en otro caso} \end{cases}$$

Donde k es el identificador de la hormiga actual, i es el arco en el que ésta se encuentra y N_i es el conjunto de arcos que aún no han sido visitados por la hormiga. τ_{ij} representa la feromona de la ruta más corta entre los arcos i y j, mientras que η_{ij} es el recíproco de la distancia más corta entre los arcos i y j. El parámetro β determina el efecto relativo de τ_{ij} contra η_{ij}, además, debe ser mayor a 0. q es una variable aleatoria uniforme

en el intervalo [0, 1] y q_0 es un valor constante predefinido que se encuentra en el intervalo [0, 1], en los experimentos realizados tomará un valor de 0.9. Cada vez que se va a escoger un arco se toma un valor para q, si $q \leq q_0$, el arco que se escoge es aquel cuyo producto entre la feromona y el recíproco de la distancia más corta es el mayor. Si $q \geq q_0$, se escoge el arco de acuerdo a una variable aleatoria donde cada arco obtiene su probabilidad según la función de distribución de probabilidad que se encuentra en S. El parámetro q_0 determina la importancia relativa de la explotación versus la exploración.

La fórmula de actualización local de feromona la aplica cada hormiga inmediatamente después de recorrer la ruta más corta entre el arco i y j mientras construye una solución, la cual se representa con la siguiente fórmula:
$\tau_{ij}^{nueva} = \rho \cdot \tau_{ij}^{previa} + (1-\rho) \cdot \tau_0 \quad \text{si}(i,j) \in T_k$
Donde T_k representa la ruta construida hasta ese punto por la hormiga k, τ_0 es el valor inicial de feromona y ρ representa el coeficiente de evaporación de la feromona.

Además de la actualización local de feromona, esta variante de ACO tiene una actualización global de feromona llevada a cabo por la mejor ruta local y global, esto con el fin de balancear la explotación de la mejor ruta obtenida por las hormigas y la exploración de nuevas rutas para encontrar una mejor. La fórmula es la siguiente:
$\tau_{ij}^{nueva} = \rho \cdot \tau_{ij}^{previa} + (1-\rho) \cdot \Delta\tau_{ij}$ donde:

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{L_b} & \text{si}(i,j) \in T_b \\ \frac{1}{L_s} & \text{si}(i,j) \in T_s \\ 0 & \text{en otro caso} \end{cases}$$

T_b y T_s representan la mejor solución global y local respectivamente; por otro lado, L_b y L_s representan la distancia obtenida por la mejor solución global y local. La búsqueda local es un proceso utilizado para mejorar la calidad de una solución. Existen varios tipos de búsqueda local; en este trabajo se va a utilizar la técnica de 2-óptimo (2-opt, como se conoce comúnmente en inglés) la cual consiste en intercambiar el orden de visita de los arcos para evaluar si es más conveniente realizarlo de esa forma o de la manera original. Este proceso se repite hasta que ya no se encuentra una mejora y solo se va a llevar a cabo en la ruta de cada camión, es decir, el proceso sobre la ruta de un vehículo es independiente de los demás. El nombre de 2-opt procede de la idea detrás del algoritmo, que consiste en retirar dos arcos de la ruta y sustituirlos por otros en búsqueda de una mejora. La Figura 4.7 ejemplifica de manera simple el proceso de intercambio llevado a cabo. El Algoritmo 2 describe el proceso realizado por la

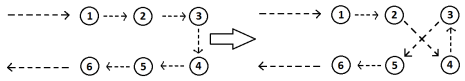

Figura 4.7: Intercambio realizado por 2-opt

búsqueda local en cada ruta, también muestra la función *2optSwap*, la cual, realiza el intercambio del orden en la ruta. La función *calculaDistancia* toma una ruta y calcula

la distancia recorrida agregando el trayecto del depósito al inicio y del final al depósito. La función *reverse* toma una lista y la devuelve, invirtiendo el orden de sus entradas.

Algoritmo 2 2-opt

```
1:  procedure 2-OPT
2:      n = número de arcos disponibles a intercambiar
3:      repeat
4:          distancia = calculaDistancia(rutaActual)
5:          for i ← 1, n−1 do
6:              for k ← i+1, n do
7:                  nuevaRuta = 2OPTSWAP(rutaActual, i, k)
8:                  nuevaDistancia = calculaDistancia(nuevaRuta)
9:                  if nuevaDistancia < distancia then
10:                     rutaActual = nuevaRuta
11:     until No mejora
12: function 2OPTSWAP(rutaActual, i, k)
13:     inicio = rutaActual[0 : i−1]
14:     cambio = reverse(rutaActual[i : k])
15:     final = rutaActual[k+1 :]
16:     return inicio + cambio + final
```

4.3 Benchmarks[1]

Después de realizar la implementación de la técnica, es necesario verificar que tenga un buen funcionamiento. Para esto utilizaremos dos de los *benchmark* más conocidos para el problema CARP. Los *benchmark* representan un conjunto de instancias del problema, los cuales, ya tienen una solución óptima conocida o un mínimo conocido, según sea el caso. La razón por la cuál es importante el uso de un *benchmark* es que permite verificar el desempeño del algoritmo, brindando información sobre el comportamiento del mismo bajo ciertos parámetros. Para esto se utilizarán tres instancias del set de *benchmark* KSHS, que corresponden a las instancias 1, 2 y 6.

La solución óptima para cada uno de ellos es conocida, por lo que resultan ideales para comparar resultados con aquellos obtenidos por la técnica elegida; las distancias óptimas se muestran en la Tabla 4.1. Además, se realizará una prueba con la instancia egl-s4-C perteneciente al *benchmark* EGL, ya que el tamaño de esta instancia (140 vértices y 190 aristas) es más parecido al problema real. Cabe destacar que no se conoce la solución óptima de la última instancia; sin embargo, tenemos un mínimo conocido que es el que se utilizará para el análisis de los resultados.

Para realizar la verificación se realizaron distintas configuraciones de parámetros

[1] El término *benchmark* se refiere a un estándar de comparación.

Tabla 4.1: Soluciones óptimas por instancia

Instancia	Solución óptima
kshs1	14661
kshs2	9863
kshs6	10197

(β, ρ, número de hormigas e iteraciones); por cada configuración se realizó un experimento que consistió en ejecutar el algoritmo 30 veces para las instancias de KSHS. Una vez que se obtuvieron los resultados, se elige la mejor configuración para utilizarla en la instancia perteneciente al *benchmark* EGL y realizar un experimento ejecutando el algoritmo 30 veces (solo se utilizó una configuración debido a la magnitud de la instancia), esto con el fin de obtener una muestra de buen tamaño para realizar un análisis de los resultados. Los valores propuestos para cada parámetro se presentan en la Tabla 4.2.

Tabla 4.2: Valores propuestos por parámetro

Parámetros	Valores
β	0.1, 1, 3, 5
ρ	0.1, 0.2, 0.3, 0.5
número de hormigas	10, 25, 100
número de iteraciones	100, 200, 400

4.3.1 Resultados del benchmark

Después de realizar las ejecuciones de las tres instancias antes mencionadas se registraron los resultados para cada una de las 144 configuraciones; debido a la gran cantidad de resultados, únicamente se registraron las veinte mejores configuraciones en una tabla. El criterio de ordenamiento fue la distancia mínima obtenida en alguna de sus 30 ejecuciones. Los valores que se tabularon son el mínimo, máximo, promedio y desviación estándar de las soluciones obtenidas en el experimento correspondiente. Además, se anexa la gráfica de la evolución de la solución para la mejor ejecución.

Benchmark kshs1

Esta instancia es la más pequeña del set KSHS ya que solo cuenta con ocho vértices y 15 aristas. La restricción de capacidad por vehículo para esta instancia es de 150, la distancia total de todas las aristas es de 8705 y se requieren cuatro vehículos para resolver este problema.

Las veinte mejores configuraciones, de acuerdo a la calidad de las soluciones obtenida en ellas se presentan en la Tabla 4.3.

Tabla 4.3: Resultados por configuración en la instancia kshs1

Configuración	β	ρ	hormigas	iteraciones	mínimo	máximo	promedio	desviación estándar
1	3	0.1	100	100	14661	16000	15433	328.66
2	3	0.1	25	200	14661	16000	15455	355.18
3	3	0.2	100	400	14661	15580	15166	214.93
4	3	0.2	25	200	14661	15972	15531	278.16
5	3	0.3	100	100	14661	15737	15442	215.24
6	3	0.3	100	400	14661	15378	15036	207.51
7	3	0.3	10	100	14661	16414	15834	402.90
8	3	0.5	100	200	14661	15593	15253	234.95
9	3	0.5	100	400	14661	15471	15065	235.75
10	5	0.1	100	200	14661	15856	15297	279.51
11	5	0.1	100	400	14661	15539	15213	253.03
12	5	0.2	100	200	14661	15549	15092	273.06
13	5	0.2	100	400	14661	15481	15092	255.50
14	5	0.3	100	400	14661	15549	15195	314.88
15	5	0.3	10	400	14661	16122	15440	338.22
16	5	0.5	100	100	14661	16000	15457	217.24
17	5	0.5	100	400	14661	15473	15218	269.60
18	3	0.3	10	200	14729	16068	15635	332.37
19	3	0.3	10	400	14729	16000	15392	278.12
20	5	0.1	25	200	14729	16061	15524	360.83

En la tabla podemos apreciar que el óptimo fue alcanzado en 17 configuraciones distintas (11.8 % del total de configuraciones), y se aprecia que la configuración 6 presenta la menor desviación estándar y promedio de todas; esto indica que además de lograr alcanzar el óptimo, tuvo el mejor comportamiento en cuanto a la calidad de soluciones encontradas. Al comparar con la siguiente mejor configuración (configuración 3), se puede observar que a pesar de que ρ disminuyó en 0.1, se obtuvo prácticamente el mismo comportamiento. Sin embargo, podemos ver que al disminuir el número de hormigas, las soluciones comienzan a tener más distancia entre ellas, afectando al promedio y por ende, a la desviación. Esto es debido a que al reducir el número de hormigas, estamos limitando la capacidad de exploración del algoritmo en el grafo.

En general, las mejores configuraciones tienen un valor para β de 3 y ρ de 0.3, pero dependen enormemente del número de hormigas no solo para encontrar una buena solución, sino para que el promedio de las soluciones disminuya y así obtener soluciones de una mejor calidad. En la Figura 4.8, se muestra la evolución de la mejor solución obtenida utilizando la configuración 6. En ella se puede observar que en las primeras iteraciones de la ejecución es cuando la distancia obtenida disminuye de manera precipitada y alcanza un punto donde ya no se obtiene una mejora hasta varias iteraciones después, siendo la iteración 275 cuando se obtiene la distancia óptima de 14661.

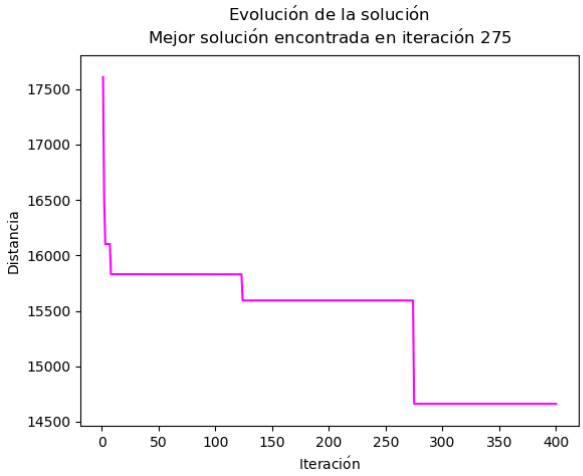

Figura 4.8: Evolución de la mejor solución para la configuración 6

Benchmark kshs2

Esta instancia cuenta con 10 vértices y 15 aristas. La restricción de capacidad por vehículo ahora es de 150. La distancia total de todas las aristas es de 6728 y se requieren cuatro vehículos para resolver este problema. Las veinte mejores configuraciones, de acuerdo a la distancia mínima obtenida en ellas se presentan en la Tabla 4.4.

A diferencia del ejemplar kshs1, el óptimo solo fue alcanzado en tres configuraciones distintas (2.08 % del total de configuraciones). Tal y como se observa en la tabla, la configuración 2 obtuvo la distancia óptima, sin embargo la configuración 9 obtuvo la mejor desviación estándar. Todas las configuraciones que lograron obtener el óptimo utilizaron un valor de $\beta = 3$ y 100 hormigas. Por otro lado, la configuración 9 tuvo un muy buen desempeño aún sin haber conseguido la distancia óptima (tan solo por una diferencia de 60), obtuvo el mínimo de los máximos y el promedio de las soluciones es el segundo mejor, por lo que también se considera una buena configuración.

En la Figura 4.9, se muestra la evolución de la mejor solución encontrada utilizando la configuración 2 ya que obtuvo el resultado óptimo y además, presenta la menor desviación estándar. Esta ejecución encontró el resultado óptimo en la iteración 39 y solo hubo dos variaciones en su evolución.

Tabla 4.4: Resultados por configuración en la instancia kshs2

Configuración	β	ρ	hormigas	iteraciones	mínimo	máximo	promedio	desviación estándar
1	3	0.1	100	200	9863	10192	10029	79.34
2	3	0.1	100	400	9863	10123	9978	52.02
3	3	0.2	100	400	9863	10192	9969	57.15
4	1	0.1	100	400	9893	10895	10430	272.92
5	1	0.2	100	400	9923	10976	10411	323.18
6	3	0.1	25	200	9923	10192	10061	103.19
7	3	0.3	100	200	9923	10192	10047	103.21
8	3	0.5	100	100	9923	10192	10083	82.26
9	5	0.1	100	400	9923	10123	9976	45.58
10	5	0.2	100	400	9923	10192	10041	98.52
11	5	0.3	25	400	9923	10609	10156	175.41
12	1	0.3	100	400	9953	10895	10461	283.29
13	1	0.5	100	400	9953	10791	10448	271.13
14	1	0.5	25	200	9953	11356	10767	360.77
15	3	0.1	100	100	9953	10192	10093	89.14
16	3	0.1	10	100	9953	10609	10163	148.50
17	3	0.1	10	400	9953	10192	10052	89.29
18	3	0.1	25	100	9953	10651	10133	133.35
19	3	0.1	25	400	9953	10192	10019	78.99
20	3	0.2	100	100	9953	10379	10105	125.61

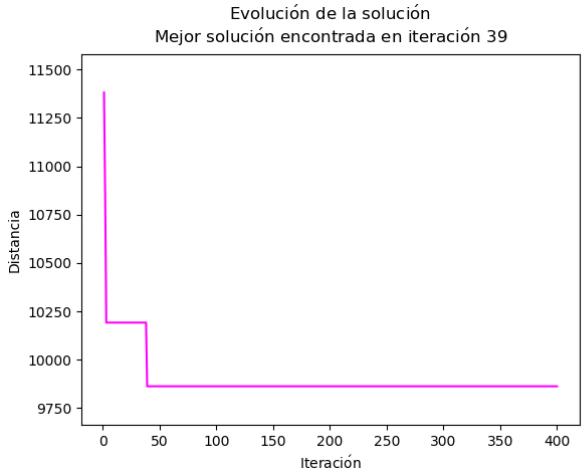

Figura 4.9: Evolución de la mejor solución para la configuración 2

Benchmark kshs6

Esta instancia cuenta con 9 vértices y 15 aristas. La restricción de capacidad por vehículo para esta instancia es de 150. La distancia total de todas las aristas es de 8525 y se requieren tres vehículos para resolver este problema. Las veinte mejores configuraciones, de acuerdo a la distancia mínima obtenida en ellas se presentan en la Tabla 4.5.

Tabla 4.5: Resultados por configuración en la instancia kshs6

Configuración	β	ρ	hormigas	iteraciones	mínimo	máximo	promedio	desviación estándar
1	3	0.1	100	400	10197	11285	10889	277.98
2	1	0.2	10	400	10305	12063	11624	375.89
3	3	0.1	10	400	10305	11355	11073	255.36
4	3	0.1	25	400	10305	11355	10951	320.62
5	3	0.3	100	200	10305	11199	11016	189.36
6	3	0.3	100	400	10305	11199	10928	223.14
7	3	0.5	100	200	10305	11199	11042	154.89
8	5	0.1	100	400	10305	11091	10971	209.16
9	5	0.1	25	400	10305	11177	11024	163.85
10	5	0.2	100	400	10305	11091	10952	185.47
11	5	0.3	100	200	10305	11199	11047	182.83
12	5	0.3	100	400	10305	11091	11039	151.65
13	1	0.1	10	400	10461	12279	11747	400.24
14	1	0.2	25	200	10461	12345	11686	335.87
15	1	0.5	100	200	10461	11733	11349	313.00
16	1	0.5	25	100	10461	12324	11731	350.14
17	3	0.1	100	100	10461	11355	11151	180.73
18	3	0.1	100	200	10461	11355	11060	192.96
19	3	0.2	100	400	10461	11188	11027	163.16
20	3	0.3	100	100	10461	11355	11111	184.43

En esta instancia, el óptimo solo fue alcanzado en la configuración 1 (0.69 % del total de configuraciones) y además, se tuvo el menor rendimiento del algoritmo. La segunda mejor solución está 100 unidades por encima del óptimo y se obtuvo en 11 configuraciones distintas. El mejor promedio de todas corresponde a la configuración 1 pero la mejor desviación estándar fue para la configuración 12, cuyo promedio es el noveno mejor. Nuevamente, el óptimo fue alcanzado con $\beta = 3$, $\rho = 0.1$, 100 hormigas y 400 iteraciones. En la Figura 4.10, se muestra la evolución de la mejor solución encontrada utilizando la configuración 1 y la iteración en la que fue encontrado el óptimo.

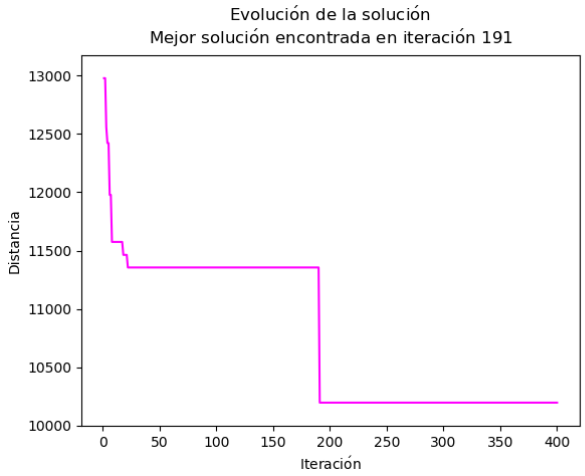

Figura 4.10: Evolución de la mejor solución para la configuración 1

A partir de los resultados anteriores, se observa que la mejor configuración es $\beta = 3$, $\rho = 0.1$, 100 hormigas y 400 iteraciones. Esta configuración fue utilizada en la instancia de egl-s4-C, que cuenta con 140 vértices y 280 aristas; el mínimo conocido para esta instancia es de 20179 [5]. En la Tabla 4.6 se observa los mínimos reportados por otros autores y en la Tabla 4.7 se puede apreciar el mínimo, máximo, promedio y desviación estándar obtenidos al utilizar la configuración elegida. En esta instancia no se

Tabla 4.6: Mínimos reportados para la instancia egl-s4-C

Belenguer Benavent 2003	Bode Irnich 2012	Longo et al. 2006	Bode Irnich 2012	Martinelli et al. 2011	Bode Irnich 2013
20179	20340	20375	20376	20380	20406

Tabla 4.7: Resultados en la instancia egl-s4-C

Configuración	β	ρ	hormigas	iteraciones	mínimo	máximo	promedio	desviación estándar
1	3	0.1	100	400	21204	21778	21516	157.67

logra alcanzar el óptimo en ninguna ejecución. A pesar de que la configuración elegida fue la mejor en el caso de las instancias KSHS, en este caso no tuvo un desempeño similar, esto en gran medida por el aumento tan repentino del espacio de búsqueda y por el número de iteraciones tan reducido para este espacio. Sin embargo, los resultados obtenidos sí se acercaron a los reportados en la literatura, por lo que se puede decir que el algoritmo funciona adecuadamente. La evolución de la solución en la ejecución que obtuvo la menor distancia se muestra en la Figura 4.11.

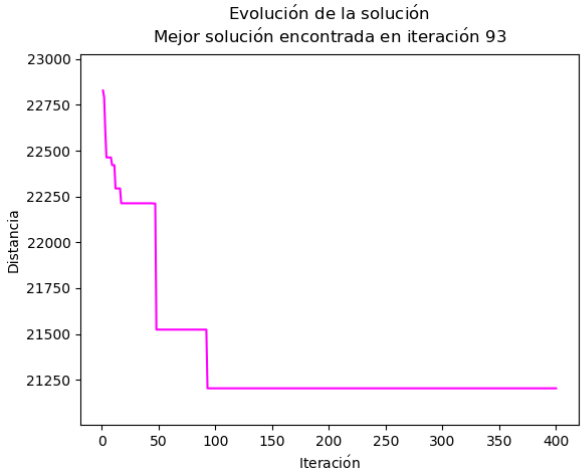

Figura 4.11: Evolución de la mejor solución

4.3.2 Análisis de resultados

En todas las instancias del *benchmark* KSHS, se obtuvo el óptimo cuando se tenían los siguiente valores: $\beta = 3$, $\rho = 0.1$ y 100 hormigas. En la instancia 1, fueron necesarias 400 iteraciones para obtener el resultado óptimo, y en las otras dos, bastaron 100 iteraciones para encontrarlo. Debido a que el mejor valor de β es dependiente de la técnica y del problema, no podemos asegurar que el obtenido en estos experimentos sea el valor ideal; en este trabajo se utilizaron los valores propuestos en la literatura. Por otro lado, ρ sí tiene un efecto conocido ya que altera la evaporación de la feromona. En nuestro caso, los mejores resultados se obtuvieron con el valor de ρ antes mencionado; otras configuraciones donde se obtuvo el óptimo y con un valor distinto de ρ presentan una mayor desviación estándar, lo que equivale a un comportamiento menos ordenado por parte de las hormigas. Esto debido a que la evaporación de la feromona fue más rápida y por ende, hubo menos convergencia hacía una misma solución, ocasionando una mayor exploración.

En el caso de la instancia egl-s4-C, los resultados fueron buenos aunque, debido a que el espacio de búsqueda de la instancia es muy grande en comparación a las otras instancias, seguramente al incrementar el número de iteraciones y/o de hormigas mejoraría la calidad de las soluciones, acercándose aún más a los resultados reportados por otros autores.

Los autores originales de la técnica en la que se basó el algoritmo presentado utilizan tres técnicas de búsqueda local en conjunto, hecho por el cual las soluciones que obtienen resultan muy buenas en tan pocas iteraciones, reportando que 150 iteraciones son suficientes para resolver los benchmarks. Por otro lado, el algoritmo que presen-

tamos es más sencillo y requiere de 400 iteraciones (quizá más) para obtener buenos resultados debido a que solo utilizamos una técnica de búsqueda local, por lo que se debe nivelar esa falta de mejora continua en las soluciones encontradas con una mayor exploración del espacio de búsqueda.

Como conclusión final, el algoritmo que utilizamos puede atacar sin problema instancias del CARP, aunque será necesario la experimentación sobre cada instancia si se desea obtener una configuración paramétrica ideal para cada una.

4.4 Configuraciones

A partir de los resultados obtenidos en la sección de *benchmarks*, se eligió una configuración de parámetros que demostró su eficacia en todas las instancias en las que se probó. Dicha configuración será utilizada para la aplicación de la técnica de optimización al problema de recolección de basura en la colonia Villa Milpa Alta. Se realizarán 30 ejecuciones utilizando la configuración $\beta = 3$, $\rho = 0.1$ y 100 hormigas; cada ejecución contará con 400 iteraciones debido a la dimensión del problema. Por cada ejecución, se guardará el mejor resultado obtenido y una vez concluidas todas las iteraciones, se obtendrá el mínimo, máximo, promedio y desviación estándar de dichos valores, esto con el fin de centrar la atención en las mejores soluciones encontradas. En la siguiente sección se describirán algunas de las restricciones tomadas en cuenta para la obtención de los resultados.

4.5 Restricciones

Al inicio de este trabajo se mencionaron algunas restricciones correspondientes al problema a resolver. Aunado a ellas, existen otras restricciones en el grafo que afectan directamente al problema y a la forma en la que se resuelve el mismo.

- Los arcos preservan el sentido original de las calles, es decir, importa el orden en el que se recorran los vértices.
- Como consecuencia de la regla anterior, una calle de doble sentido se convierte en dos calles, cada una con un sentido.
- Existe un vértice utilizado como depósito, de él partirán todos los camiones; también será el destino de los mismos una vez que agoten su capacidad de carga. Este punto se ubicó lo más cercano posible a la estación de transferencia.
- Todos los arcos deben ser visitados al menos una vez por alguno de los camiones recolectores.

- El grafo se ajusta al mapa descargado, por lo que pueden existir diferencias con el estado actual de las calles de Villa Milpa Alta.
- El cálculo de la basura generada diariamente en la región se obtuvo a partir de la cantidad de residuos por persona al día en la delegación Milpa Alta, multiplicada por el número de habitantes de la colonia.
- La cantidad de basura obtenida en el punto anterior se distribuyó de manera uniforme sobre la distancia, esto con el fin de asignar una cierta cantidad de residuos a cada arco.
- El número de camiones a utilizar es de tres, esto debido a los cálculos realizados para encontrar la cantidad de basura producida diariamente en la zona.
- La capacidad de carga de los camiones se fijó en seis toneladas.
- No se cuenta con las rutas verdaderas que se utilizan para la recolección de residuos actualmente, este trabajo presentará una propuesta.

De estas restricciones, las que más impacto tienen sobre el problema son la primera y segunda ya que se incrementa un poco más el espacio de búsqueda. Por ejemplo, si se tiene una calle de doble sentido, el camión tendrá que pasar por ambos sentidos para poder marcarla como recorrida. Para poder representar de manera fiel a la realidad el problema de la recolección, se necesitarían muchas más restricciones; es importante recordar que entre menos restricciones se impongan al problema, será menos complicado atacarlo.

Capítulo 5

Resultados

Tomando en cuenta las restricciones presentadas en capítulos anteriores y utilizando la configuración obtenida gracias a los *benchmarks*, se obtuvieron los resultados de la Figura 5.1. Cabe destacar que la distancia obtenida es la suma de las distancias recorridas por cada camión, incluyendo salida y regreso al depósito.

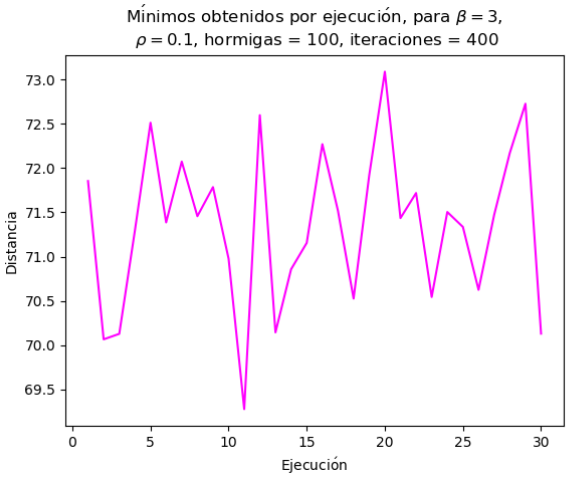

Figura 5.1: Mejores resultados por ejecución para el problema de Villa Milpa Alta

Los resultados exactos obtenidos después de ejecutar el algoritmo (truncados a tres decimales) se muestran en la Tabla 5.1, mientras que en la Tabla 5.2 se muestra el máximo, promedio y desviación estándar de los mejores resultados.

Podemos observar que los resultados arrojan distancias de entre 69 y 72 km por lo que la máxima diferencia encontrada entre ellas es de 3 km aproximadamente; esto demuestra que el algoritmo tuvo un comportamiento similar en todas las ejecuciones.

La mayoría de los resultados obtenidos se encontró en la franja de 70.5 a 72.5 km, lo que puede indicar que debajo de esa distancia empieza a ser más difícil encontrar una solución y/o que el algoritmo se estancó en esa región. Sin embargo, se obtuvo un resultado de 69.27 km en la iteración 100 de la ejecución 11; en la gráfica se identifica de manera sencilla pues es el pico que se encuentra en la parte más baja.

Este mejor resultado no está tan alejado del promedio de soluciones obtenidas por el algoritmo, sin embargo, es el único que se encontró por debajo de los 70 km. La desviación estándar obtenida indica que la configuración utilizada hace que el comportamiento de las soluciones sea estable; esto es bueno ya que podemos esperar soluciones dentro del mismo rango y además hace notar el trabajo de la colonia de hormigas, alejándose de un comportamiento al estilo de búsqueda aleatoria.

Tabla 5.1: Mejores resultados por ejecución para el problema de Villa Milpa Alta

Ejecución	Mejor distancia obtenida (km)	Ejecución	Mejor distancia obtenida (km)
1	71.853	16	72.270
2	70.065	17	71.517
3	70.127	18	70.526
4	71.293	19	71.918
5	72.512	20	73.089
6	71.386	21	71.435
7	72.073	22	71.718
8	71.456	23	70.544
9	71.785	24	71.504
10	70.976	25	71.335
11	69.277	26	70.624
12	72.598	27	71.472
13	70.144	28	72.170
14	70.857	29	72.727
15	71.155	30	70.130

Con respecto a la evolución de la solución durante el proceso del algoritmo, el mejor resultado se obtuvo en la iteración 100 de la ejecución correspondiente; de la iteración 1 a la 100 tuvo un muy buen desempeño pues mejoró en varias ocasiones la calidad de la solución. El hecho de que no haya podido mejorar la solución en 300 iteraciones más puede ser por varias razones, entre ellas las más probables son: la exploración que realizaron las hormigas no fue exhaustiva o se encontró un mínimo local (incluso la solución óptima). Este comportamiento se puede apreciar en la Figura 5.2.

Debido a la complejidad del problema, no tenemos un medio para decir que las solu-

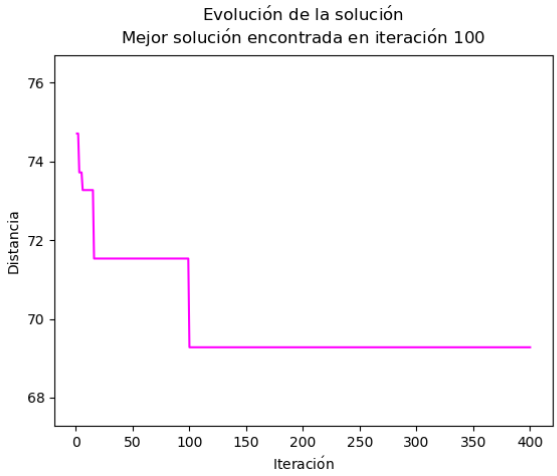

Figura 5.2: Evolución de la solución en el problema de Villa Milpa Alta

ciones encontradas son buenas o malas, mucho menos podemos hablar de un óptimo ya que sería necesario obtener todas las soluciones posibles para poderlas comparar entre sí. Debido a que las rutas de recolección no se nos proporcionaron, no es posible realizar una comparación con los resultados descritos anteriormente.

Tabla 5.2: Datos estadísticos de los resultados obtenidos

β	ρ	hormigas	iteraciones	mínimo	máximo	promedio	desviación estándar
3	0.1	100	400	69.2774800751	73.0890560634	71.3516709932	0.90

Dentro de la mejor solución obtenida también podemos analizar la distancia a recorrer por cada camión. En este caso se obtuvieron los siguientes valores: 25.949, 26.481 y 16.486 km. Para mostrar las rutas generadas, se graficó el mapa de la colonia Villa Milpa Alta y se dibujó la ruta sobre sus calles; cada calle está representada por una flecha de color negro que indica el sentido de circulación, las flechas azules y rojas pertenecen al recorrido del camión, el color rojo indica que se realizó el servicio de recolección en esa calle y el color azul indica que el camión transitó por esa calle pero no recogió basura. El depósito se encuentra en la parte inferior derecha de cada imagen y corresponde al vértice 107.

Algo que resulta interesante es que el algoritmo dividió el mapa en dos partes, ya que la primer ruta brinda el servicio de recolección en la zona norte y oeste principalmente, mientras que la segunda ruta da servicio a la parte sur y este. La última ruta solo cubre algunas calles a las cuales no se les dio servicio en las rutas anteriores, enfocándose principalmente en la zona noreste del mapa; las primeras dos rutas recorren

casi la misma distancia mientras que la tercera solo es para terminar el trabajo que no pudieron hacer las otras. Este comportamiento se observa en las Figuras 5.3, 5.4 y 5.5. Otro punto importante al analizar las figuras es que los camiones no dan servicio en algunas calles, esto resulta poco intuitivo de primera mano ya que la mayoría de las personas optaría por una estrategia glotona o greedy (recogiendo basura por todas las calles que transite hasta llenar su capacidad) al momento de crear rutas de recolección. Esto también tiene un impacto en la implementación de este trabajo en la vida real ya que los habitantes tendrían que identificar de algún modo el camión que les corresponde.

La forma en la que se construyó el algoritmo y en que se codificó el grafo no permiten que se agreguen camiones adicionales a las soluciones, ya que se trabaja con el número de camiones justo para la cantidad de residuos que genera la colonia. En caso de que se quisiera utilizar un número menor de camiones, uno o varios de ellos tendrían que cubrir más de una ruta. Además de esto, el hecho de que tengamos las calles de doble sentido como dos arcos con direcciones opuestas hace que en algunos momentos el algoritmo opte por hacer una vuelta en u, lo que sería poco viable en el problema real. Se necesitan más datos sobre el mapa y restricciones sobre el grafo para lograr evitar al 100 % este tipo de comportamientos.

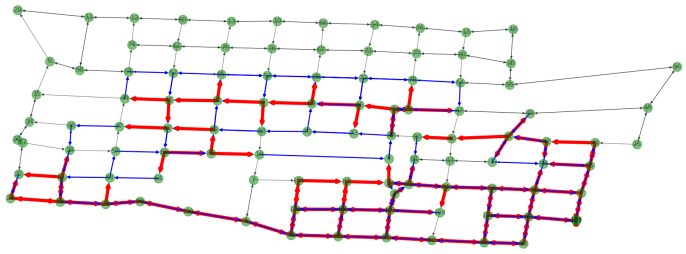

Figura 5.3: Primer ruta generada

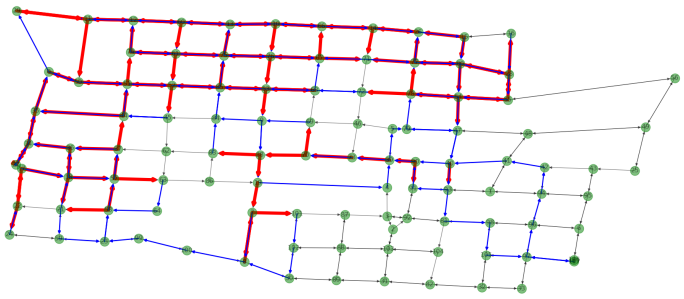

Figura 5.4: Segunda ruta generada

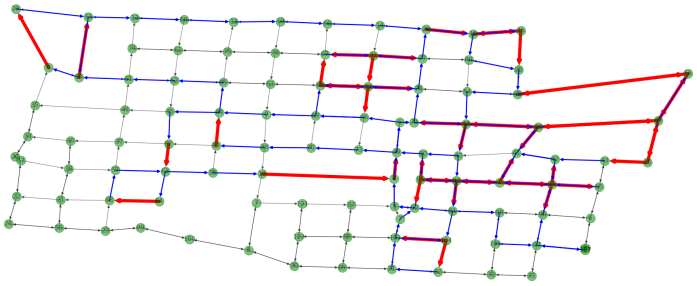

Figura 5.5: Tercer ruta generada

Conclusiones

La problemática de la recolección de residuos en la Ciudad de México aumenta año con año, esto hace que cada vez sea más difícil encontrar soluciones factibles que se puedan llevar a cabo para atacar de manera efectiva el problema. Uno de los puntos clave en la recolección es la generación de rutas, las cuales se siguen obteniendo a través del criterio y experiencia de los operadores de los vehículos de recolección. Este trabajo tuvo como uno de sus objetivos presentar una propuesta para optimizar la creación de rutas de recolección tomando como caso de estudio la colonia Villa Milpa Alta, de la delegación que lleva el mismo nombre.

En el proceso de optimización de rutas de recolección, la obtención del grafo para aplicar la técnica de optimización tuvo varias dificultades y limitaciones. En primer lugar no existe una plataforma gubernamental para descargar los datos actualizados de la zona geográfica sobre la cual se desarrolló el tema, por lo que, se tuvo que recurrir a otras alternativas que presentaban algunos datos erróneos, incompletos y desactualizados. Otra dificultad fue que una vez que se obtuvieron los datos fue necesario ajustarlos a las necesidades de nuestro problema. Este proceso tomó mucho tiempo debido a que se tenía que corroborar los datos calle por calle, con el fin de que el grafo estuviera lo más apegado a la realidad como fuera posible.

A pesar de la gran cantidad de artículos referentes al problema CARP, pocos son los que explican de una manera detallada la implementación de la técnica y el procedimiento que siguieron para resolver su problema. Además, la dificultad del problema CARP hace que muchas de las técnicas utilizadas por otros autores fueran muy complejas de implementar. La elección de la variante de ACO utilizada en este trabajo se debió a que era la única que no solo mostraba buenos resultados en problemas similares al de este trabajo, sino que además, la claridad de su explicación facilitó la puesta en práctica.

Al observar los resultados de los *benchmarks* se comprobó la calidad de la técnica elegida y se encontró una configuración que funcionó de buena manera en el problema real. En especial la instancia egl-s4-C fue de gran ayuda ya que sus dimensiones eran similares a las del problema a resolver y, a pesar de no tener una solución óptima registrada, tenía varios resultados obtenidos por otros autores, los cuales no estuvieron

tan alejados de los valores que se obtuvieron con este programa.

Para poder comparar de manera adecuada los resultados obtenidos en el problema real sería necesario contar con las rutas utilizadas actualmente en la colonia Villa Milpa Alta. Sin embargo, la delegación no nos brindó la información requerida, por lo que uno de los objetivos de este trabajo no se pudo cumplir, afectando el apartado de resultados ya que no se pudo comparar las rutas obtenidas con las actuales.

Con respecto a las configuraciones, sería bueno realizar distintas variaciones de parámetros, tal y como se hizo en los *benchmarks*, para obtener un conjunto de resultados y poderlos analizar y comparar entre sí, con el fin de observar la influencia de cada parámetro en la calidad de la solución. Esto también ayudaría a determinar si existe alguna configuración que brinde un mejor desempeño en este problema, ya que es importante recordar que no existe una mejor configuración para todos los problemas, sino que el desempeño de cada una depende del mismo. En problemas de gran magnitud, como el nuestro, se requiere de mucho tiempo y poder computacional para probar varias configuraciones, siendo esto una enorme limitante para realizar las comparaciones antes mencionadas.

El método de recolección de basura actual, no permite aplicar técnicas de optimización de una manera sencilla, ya que nos obliga a tomar en cuenta casi todas las calles de una determinada región. En otros trabajos se hace hincapié en la reducción del espacio del problema debido al punto antes mencionado; para poder hacer algo similar en la ciudad de México sería necesario realizar algunos cambios al sistema actual de recolección. Uno de ellos podría ser establecer horarios fijos de servicio a ciertas calles, esto provocaría que el problema se reduzca solo a las calles a las que se dará servicio y no a toda la colonia, reduciendo la complejidad. Otra propuesta sería el uso de contenedores donde las personas depositen su basura, así, los camiones solo tendrían que pasar a los contenedores en vez de recorrer todas las calles. También se podrían establecer regiones geográficas dentro del área a la que se dará servicio y asignar vehículos a cada zona, con el fin de aplicar la técnica de optimización a cada zona en vez de aplicarla a toda la región, convirtiendo cada zona en un subproblema del problema original, disminuyendo así su complejidad.

El proceso de cambiar el modelo de recolección actual por otro más optimizable, no será fácil ya que se necesitará nueva infraestructura y disposición por parte de la población para llevarlo a cabo. Sin embargo, esta propuesta debe ser tomada en cuenta por las autoridades gubernamentales debido al continuo aumento poblacional y por ende, a una mayor generación de residuos. Este trabajo muestra que la ciencia se puede aplicar en la solución de problemas complejos, que se presentan cotidianamente en la ciudad.

Apéndice A

Ruta generada en la colonia Ignacio Zaragoza

En este apartado se incluyen las rutas generadas para otra zona de la Ciudad de México, la colonia Ignacio Zaragoza perteneciente a la delegación Venustiano Carranza. Se utilizó la metodología descrita a lo largo de este trabajo, por lo que además de obtener resultados para la colonia de Villa Milpa Alta, se demuestra que el programa desarrollado es capaz de obtener soluciones para otras zonas, siempre y cuando respeten el formato y restricciones mencionadas en el capítulo de metodología.

Para generar estas rutas se consideró que el número de habitantes en la colonia era de 10175, y que en promedio, cada persona genera 1.98 kg de basura al día, dando como resultado 20146 kg de residuos generados al día en toda la colonia; en cuanto a los vehículos, se consideró la capacidad de carga en 6000 kg.

Como resultado se obtuvo una distancia total de aproximadamente 57 km utilizando cuatro vehículos; las distancias de cada ruta son: 16.07 km, 17.42 km, 13.62 km y 8.85 km, tal y como se muestran en las siguientes imágenes. Todo esto nos permite ver la utilidad de esta herramienta, pues con ella, resulta más sencillo atacar este problema y la manera de visualizarlo se simplifica en gran medida.

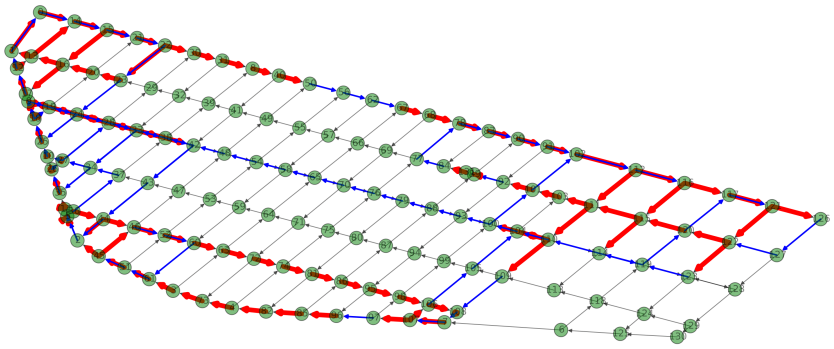

Figura 1.1: Primer ruta generada

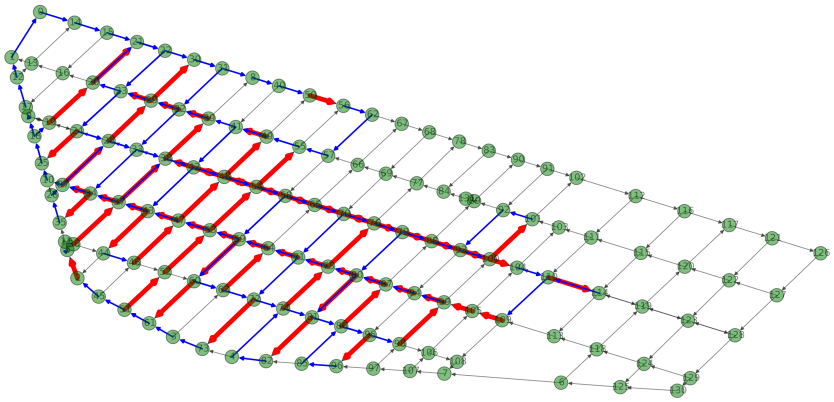

Figura 1.2: Segunda ruta generada

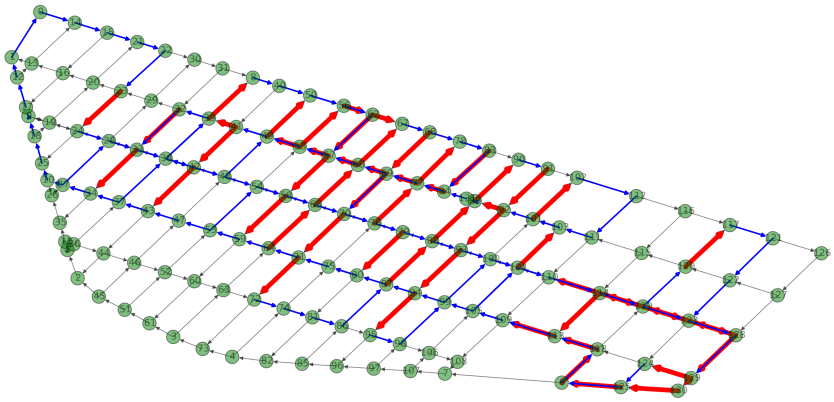

Figura 1.3: Tercer ruta generada

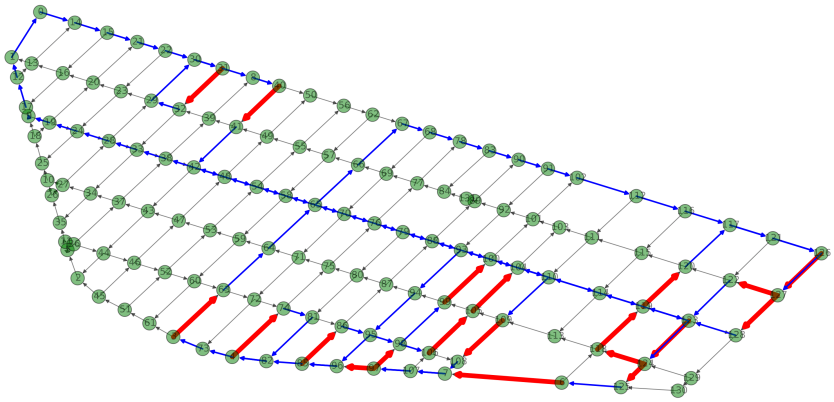

Figura 1.4: Cuarta ruta generada

Bibliografía

[1] E. Angelelli and M. G. Speranza. *The Application of a Vehicle Routing Model to a waste-collection problem: two case studies*. Palgrave Macmillan Journals, 2002.

[2] Arjang A. Assad, Wen-Lea Pearn, and Bruce L. Golden. The capacitated chinese postman problem: Lower bounds and solvable cases. *American Journal of Mathematical and Management Sciences*, 7(1-2):63–88, 1987.

[3] Alessandro Astolfi. Optimization, an introduction. 2006.

[4] Nilesh A. Aynodkar and S. M. Bhosale. Application of arc routing problem for municipal solid waste collection in kolhapur city, india. 2015.

[5] Benchmarks. `http://logistik.bwl.uni-mainz.de/benchmarks.php`. Fecha de consulta: 2018-09-01.

[6] Claude Berge. *Graphs*. Elsevier Science Publishing Company, 1970.

[7] Vašek Chvátal, William Cook, George B. Dantzig, Delbert R. Fulkerson, and Selmer M. Johnson. *Solution of a Large-Scale Traveling-Salesman Problem*, pages 7–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[8] Angel Corberán and Christian Prins. Recent results on arc routing problems: An annotated bibliography. *Networks*, 56(1):50–69, 2010.

[9] G. A. Croes. A method for solving traveling salesman problems. 1958.

[10] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Manage. Sci.*, October 1959.

[11] Gobierno de la Ciudad de México. *Inventario de Residuos Sólidos de la Ciudad de México*. Secretaría del Medio Ambiente de la Ciudad de México, 2015.

[12] Delegación Milpa Alta. Sitio web de la delegación milpa alta, 2017.

[13] Marco Dorigo. *Optimization, Learning and Natural Algorithms*. Dipartimento di Elettronica, 1992.

[14] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. The MIT Press, 2004.

[15] M. Dror. *Arc Routing - Theory, solutions and applications*. Kluwer Academic Publishers, 2000.

[16] H. A. Eiselt, Michel Gendreau, and Gilbert Laporte. *Arc Routing Problems Part II: The Rural Postman Problem*. Operations Research, 1995.

[17] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, May 2006.

[18] G. Ghiani and G. Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, pages 11–18, 2000.

[19] B. L. Golden, J. S. Dearmon, and E. K. Baker. Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research*, 10:47–59, 1983.

[20] Bruce L. Golden and Richard T. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

[21] S. Goss, S. Aron, J. Deneubourg, and J. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76(12):579–581, December 1989.

[22] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. *Self-organized shortcuts in the Argentine ant*. Naturwissenschaften, 1989.

[23] Ralph P.. Grimaldi. *Matematicas discreta y combinatoria : una introducción con aplicaciones*. México, Addison Wesley Longman, 1998. Traduccion de: Discrete and combinatorial mathematics. An applied introduction.

[24] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[25] Z Ismail and Ser Lee Loh. Ant colony optimization for solving solid waste collection scheduling problems. 5:199–205, 03 2009.

[26] Michael R. Garey; David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[27] Menger Karl. *Das Botenproblem, in: Ergebmisse eines Mathematischen Kolloquiums 2*. 1932.

[28] John R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT press, Cambridge, MA, USA, 1992.

[29] Mei-Ko Kwan. *Graphic programming using odd or even points*. 1962.

[30] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Cherif. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(1):159–185, Oct 2004.

[31] Philippe Lacomme, Christian Prins, and Alain Tanguy. *First Competitive Ant Colony Scheme for the CARP*, pages 426–427. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[32] J. K. Lenstra and A. H. G. Rinnooy Kan. On general routing problems. *Networks*, 6(3):273–280, 1976.

[33] Jan Karel Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[34] J. Liu and Y. He. Ant colony algorithm for waste collection vehicle arc routing problem with turn constraints. In *2012 Eighth International Conference on Computational Intelligence and Security*, pages 35–39, Nov 2012.

[35] Humberto Longo, Marcus Poggi de Aragao, and Uchoa Eduardo. Solving capacitated arc routing problems using a transformation to the cvrp. *Computers & Operations Research*, 33:1823–1837, 2006.

[36] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[37] H. Miiller-Merbach. *Zweimal travelling salesman*. 1983.

[38] Ibrahim H. Osman and Gilbert Laporte. Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623, Oct 1996.

[39] D. Otoo, S. K. Amponsah, and C. Sebil. Capacitated Clustering and Collection of Solid Waste in Kwadaso Estate, Kumasi. *Journal of Asian Scientific Research*, 4(8):460–472, August 2014.

[40] Alan R. Parkinson, Richard J. Balling, and John D. Hedengren. *Optimization methods for engineering design. Applications and theory*. Brigham Young University, 2013.

[41] Sriram Pemmaraju and Professor Steven Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica ®*. Cambridge University Press, 2003.

[42] F.B. Pereira and J. Tavares. *Bio-Inspired Algorithms for the Vehicle Routing Problem*, volume 161. Springer, n/a, spr edition, 2009.

[43] Ciara Pike-Burke. Mult-objective optimization.

[44] Christian Prins. *A GRASP Evolutionary Local Search Hybrid for the Vehicle Routing Problem*, pages 35–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[45] Masoud Rabbani and Setare Mohammadi. *Modelling a multi depot k-chinese postman problem with consideration of priorities for servicing arcs*. American Scientific Publishers, 2015.

[46] A. E. Rizzoli, F. Oliverio, R. Montemanni, and L. M. Gambardella. Ant colony optimisation for vehicle routing problems: from theory to applications. Technical report, 2004.

[47] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, 2010.

[48] Secretaría de Desarrollo Social. *Manual Técnico sobre generación, recolección y transferencia de residuos sólidos municipales.* Gobierno Federal, 2001.

[49] Huang Shan-Huen and Lin Tsan-Hwan. *Using Ant Colony Optimization to solve Periodic Arc Routing Problem with Refill Points*. Journal of Industrial and Production Engineering, 2014.

[50] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[51] Han-Shiuan Tsai and Ching-Jung Ting. An ant colony optimization for the capacitated arc routing problem. 2016.

[52] Unal Ufuktepe and Goksen Bacak Turan. Applications of graph coloring, 05 2005.

[53] Thibaut Vidal. Node, edge, arc routing and turn penalties: Multiple problems—one neighborhood extension. 65, 05 2017.

[54] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658 – 673, 2014.

[55] Buch von F. Voigt. Das handlungsreisendenproblem, tsp. 1831.

[56] Sanne Wohlk. *A decade of capacitated arc routing*. Aarhus University, 2008.

[57] L. N. Xing, P. Rohlfshagen, Y. W. Chen, and X. Yao. A hybrid ant colony optimization algorithm for the extended capacitated arc routing problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(4):1110–1123, Aug 2011.

[58] Weijian Xue and Kai Cao. Optimal routing for waste collection: a case study in singapore. *International Journal of Geographical Information Science*, 30(3):554–572, 2016.

[59] Haifeng Yu. *Optimization of Vehicle Routing and scheduling with travel time variability-application in winter road maintenance*. Department of civil and environmental Engineering, 2014.