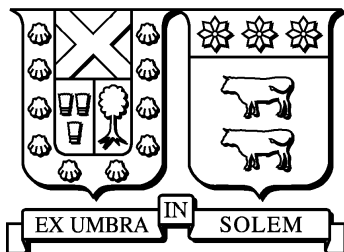


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



“APLICACIÓN DEL SISTEMA DE HORMIGAS
PARA EL PROBLEMA DE ENRUTAMIENTO DE
VEHÍCULOS DE SUMINISTROS EN DESASTRES
NATURALES”

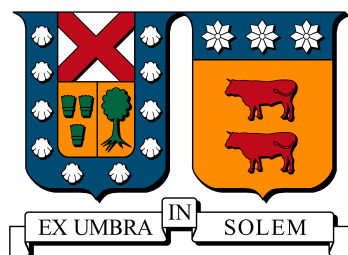
SERGIO YÁÑEZ SILVA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: ELIZABETH MONTERO

SEPTIEMBRE 2019

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



**“APLICACIÓN DEL SISTEMA DE HORMIGAS
PARA EL PROBLEMA DE ENRUTAMIENTO DE
VEHÍCULOS DE SUMINISTROS EN
DESASTRES NATURALES”**

SERGIO YÁÑEZ SILVA

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO**

PROFESOR GUÍA: ELIZABETH MONTERO

PROFESOR CORREFERENTE: JOSÉ LUIS MARTÍ

SEPTIEMBRE 2019

MATERIAL DE REFERENCIA, SU USO NO INVOLUCRA RESPONSABILIDAD DEL AUTOR O DE LA INSTITUCIÓN

Agradecimientos

Léase con voz de agradecimiento de los premios de la Academia

Terminé la memoria. Me tomó bastante tiempo y, por lo tanto, tengo mucho que agradecer.

A Kevin Feige, los hermanos Russo, Tony y Steve por todas las historias, las risas y las lágrimas.

A los profesores Diego Arroyuelo, Liubov Dombrovskaia, José Luis Martí, Cristopher Arenas, Raúl Monge, Claudio Lobos, Pedro Godoy, Luis Hevia, David Zambrano, Oscar Encina, Ricardo Ñanculef y Andrés Moreira por tener toda la paciencia del mundo con un estudiante distraído y con la cabeza en las nubes. Aprendí muchísimo siendo su alumno y en varios casos su ayudante, y soy un mejor profesional por ello ¡Muchas gracias!

A mis amigos y compañeros de carrera Felipe Suazo, Javier Santibáñez, Daniel Morales, Ignacio Tolosa, Víctor Franchis, Felipe González y Juan Pablo Castillo ¡Son lo máximo!

A mis mejores amigos Pablo y Chris, y las infaltables noches de ligas de PES que tanto extraño viviendo en Seattle ¡Gracias!

A Mary y Tomás, siempre dispuestos a leer todo lo que les mando y escuchar todo lo que les digo, siempre dándome su cariñoso feedback ¡Muchas gracias!

A mis tías, tíos, primas y primos. Es difícil mantener el contacto desde tan lejos especialmente cuando estamos todos tan metidos en nuestras propias vidas. Pero se que en sus casas siempre tendré un hogar, y el cariño que me tienen se siente todos los días del año ¡Muchas gracias por acompañarme en esta aventura!

A mis abuelos, a quienes extraño muchísimo todos los días. Siempre pendientes del “pelao”, soy muy afortunado de tenerlos a los cinco en mi vida ¡Muchas gracias!

Al Nico y al Martín, best bros, por el apoyo desde lejos y la confianza de que, aunque el mundo se esté dando vueltas por todos lados, siempre los voy a tener en mi esquina preparándome para el siguiente round ¡Muchas gracias!

A la profesora Elizabeth Montero. Me imagino que lidiar con alumnos que se mandan a cambiar al otro lado del mundo y desaparecen por varios meses del correo electrónico no es parte del cargo de profesor. Muchas gracias por todo el apoyo, por querer sacar adelante el proyecto conmigo hasta literalmente los últimos 20 minutos antes de entregarlo, por ser persistente y sacar lo mejor de mí en cada obstáculo que se me cruzaba. Esta memoria no sería lo mismo sin haber tenido una profesora tan seca y comprometida ¡Muchísimas gracias!

A mi amada Trini, mi compañera para siempre, por ser mi fan número uno en las buenas y en las malas. Durante el proceso de esta memoria nos vinimos a vivir la aventura de la vida juntos, solos, lejos de nuestras familias y amigos, pero nos tuvimos a los dos. En esta memoria está tu amor y tu cariño, tu paciencia y comprensión cuando me costaba avanzar por el estrés, cuando en la madrugada me decías que quizás mejor lo viera en la mañana con las pilas recargadas, y me ayudabas a levantarme al día siguiente ¡Amor! ¡Terminé la memoria! ¡Te amo y muchas gracias por todo!

A mis papás, por haber sido tan jugados en que cerrara el ciclo de la Universidad. Mi papá se quedaba noches leyendo sobre inteligencia artificial, sobre algoritmos de hormigas, entendiendo la matemática y la lógica detrás e interpretando los números conmigo, todo esto a la distancia. Gracias a ustedes tuve la oportunidad de estudiar lo que quería, disfrutar lo que hago y ahora vivir en Seattle trabajando en lo que me encanta ¡Lo logramos! ¡Los quiero mucho!

Este logro fue largo y pesado, pero se acabó. Muchas gracias UTFSM por ser un lugar donde pude aprender todo lo que quise, y que me entregó las herramientas necesarias para poder hacer lo que me gusta el resto de mi vida.

Hakuna Matata!

Resumen

El tiempo de respuesta y la eficiencia de los vehículos de suministros las primeras horas después de desastres naturales es un factor crítico de la mitigación de la crisis. Estas circunstancias son comúnmente modeladas como un problema de enrutamiento de vehículos con múltiples depósitos (MDVRP por sus siglas en inglés). Este problema consiste en un conjunto de puntos que deben recibir cierta cantidad de suministros, entregados a partir de un conjunto de depósitos y vehículos que deben comenzar y finalizar sus rutas en dichos depósitos, respetando limitaciones de tiempo y capacidad de los vehículos.

Este trabajo propone un algoritmo de optimización por colonias de hormigas para encontrar soluciones al problema. El algoritmo construye soluciones inspirado en el comportamiento de las hormigas, las cuales utilizan feromonas para indicar a sus pares las mejores rutas que han encontrado. Esto se aplica en la construcción de soluciones, ponderando movimientos aleatorios con información histórica contenida en estas feromonas. Luego de dicha construcción, se aplican operadores de búsqueda local para mejorar las soluciones obtenidas, y luego se depositan más feromonas para las hormigas de la iteración siguiente.

Se presenta un análisis experimental de las principales componentes del algoritmo considerando diferentes esquemas de búsqueda local como estrategias de mejor mejora versus primera mejora, y el uso de los tres operadores propuestos (CROSS-Exchange, Or-opt y Relocate) versus el uso de solo uno de ellos seleccionado al azar en cada iteración. También se comparan los esquemas de control de feromonas, analizando el desempeño de un esquema MaxMin variable contra los resultados de dos esquemas de MaxMin fijos, uno dependiente de la instancia y otro con valores fijos iguales para todos los casos. Para el análisis de consideren 15 instancias de Christofides, clásico marco de referencia para problemas MDVRP, de

entre 50 y 249 puntos de demanda.

Los resultados muestran que el algoritmo es competitivo frente a otras propuestas de la literatura, obteniendo mejores resultados que otros métodos meta-heurísticos, sin alcanzar sin embargo a los mejores resultados actuales.

Abstract

In the wake of a natural disaster, the efficiency and response time of supply vehicles is a key factor in the mitigation of the crisis. These circumstances are commonly modeled as multi-depot vehicle routing problems (MDVRP). This problem consists in a set of demand points or customers that need to be supplied by a set of depots and their vehicles, which must start and finish in these depots, and constrained by time and vehicle capacity.

This work proposes an ant colony optimization algorithm to find solutions to this problem. The algorithm builds candidate solutions inspired by ant behavior in nature, which use pheromones to indicate to their peers the best routes they have found so far. This is applied in the construction of candidate solutions, weighing random movements with historical data contained in these pheromones. After this construction, local search operators are applied to improve the solutions, and finally new pheromones are applied to communicate the results to future ants.

This work presents an experimental analysis on the principal components of the algorithm, considering different local search strategies like first improvement against best improvement, and the use of only one of the local search operators (CROSS-Exchange, Or-Opt and Relocate) chosen at random in every iteration against using all of them in every iteration. Different pheromone control strategies are also compared, analysing the performance of a variable MaxMin method against to fixed MaxMin strategies, one using different fixed boundaries depending on the instance and another using the same fixed boundaries for all instances.

Lastly, 15 instances proposed by Christofides are used, widely known as a classic benchmark in literature.

Results show that the algorithm is competitive against other algorithms from the literature, obtaining better results than some of the other meta-heuristics, but without finding the best known solutions.

Índice de Contenidos

Agradecimientos	III
Resumen	V
Abstract	VII
Índice de Contenidos	IX
Lista de Tablas	XII
Lista de Figuras	XIII
Introducción	1
1. Definición del Problema	4
1.1. Variantes del VRP Relevantes para este Trabajo	5
1.1.1. VRP Multi-Depósito	5
1.1.2. OVRP	5
1.1.3. Demanda Estocástica en Emergencias	6
1.2. Modelo Matemático	6
1.2.1. Bases del modelo y notación	6
1.2.2. Modelo matemático	8

2. Estado del Arte	10
2.1. Solución al VRP	10
2.1.1. Algunos Algoritmos Propuestos	11
2.2. Solución a las Variantes del VRP	13
2.2.1. VRP Multi-Depósito	13
2.2.2. OVRP	14
2.2.3. Problema de Enrutamiento de Vehículos para Emergencias	15
2.3. Optimización por Colonias de Hormigas	15
2.3.1. Modelo Clásico	16
2.3.2. ACO Aplicado al VRP	17
2.3.3. ACO aplicado a emergencias	19
2.4. Ajuste del algoritmo	19
3. Propuesta	21
3.1. Representación	22
3.2. Función de evaluación	23
3.3. Estructura	24
3.3.1. Inicialización	24
3.3.2. Asignación de puntos de demanda a depósitos	24
3.3.3. Estructura de cada iteración	26
3.3.4. Criterios de detención	31
3.3.5. Resumen	31
4. Experimentos y Resultados	33
4.1. Experimentación	33
4.2. Instancias de Prueba	34

4.2.1.	Equipamiento técnico	35
4.2.2.	Sintonización de Parámetros	35
4.2.3.	Detalles de la ejecución	38
4.3.	Resultados	38
4.3.1.	Esquema de Búsqueda Local	38
4.3.2.	Esquema de control de feromona	45
4.3.3.	Alternativa para operadores de búsqueda local	52
4.3.4.	Comparación con otros trabajos del estado del arte	58
5.	Conclusiones	61
	Bibliografía	64

Índice de cuadros

1.1. Notación y símbolos usados en el modelo	7
3.1. Notación y símbolos usados en el algoritmo	21
3.2. Parámetros del algoritmo	32
4.1. Instancias de Christofides	35
4.2. Parámetros sintonizados por ParamILS. Para cada parámetros se presenta su conjunto de valores posibles y valores iniciales.	36
4.3. Configuración de parámetros entregada por ParamILS	37
4.4. Comparación de trabajos del estado del arte con el algoritmo propuesto . .	59

Índice de figuras

3.1. Representación de una solución. Cada ruta es representada por una lista de tuplas punto - cantidad distribuida, cada depósito es representado por una lista de rutas, una para cada vehículo, y finalmente la solución completa es representada por una lista de cada una de las soluciones para sus depósitos.	22
3.2. Ejemplo de solución propuesta	23
3.3. CROSS Exchange en acción.	28
3.4. Or-Opt en acción.	29
3.5. Relocate en acción.	30
4.1. Comparación Búsqueda Local: Primera mejora versus Mejor Mejora – Instancias de Christofides: C1 - C4	40
4.2. Comparación Búsqueda Local: Primera mejora versus Mejor Mejora – Instancias de Christofides: C5-C8	42
4.3. Comparación Búsqueda Local: Primera mejora versus Mejor Mejora – Instancias de Christofides: C9-C12	43
4.4. Comparación Búsqueda Local: Primera mejora versus Mejor Mejora – Instancias de Christofides: C15-C18	45
4.5. Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C1-C3	46

4.6.	Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C4-C6	48
4.7.	Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C7-C9	49
4.8.	Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C10-C12	50
4.9.	Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C15-C18	51
4.10.	Comparación Operadores en Búsqueda local: Operador al azar versus todos los operadores (Mejor Mejora) – Instancias de Christofides: C1 - C4	53
4.11.	Comparación Operadores en Búsqueda local: Operador al azar versus todos los operadores (Mejor Mejora) – Instancias de Christofides: C5 - C8	55
4.12.	Comparación Operadores en Búsqueda local: Operador al azar versus todos los operadores (Mejor Mejora) – Instancias de Christofides: C9 - C12	56
4.13.	Comparación Operadores en Búsqueda local: Operador al azar versus todos los operadores (Mejor Mejora) – Instancias de Christofides: C15 - C18	57

Introducción

En el último tiempo, múltiples desastres naturales han impactado diversas regiones del planeta, afectando de manera directa a poblaciones enteras, y, en gran parte de los casos, significando grandes pérdidas humanas y materiales. Entre los años 2004 y 2014, se estima que más de 700 mil personas murieron producto de un desastre natural, y más de 1.2 billones de personas fueron afectadas ([18]). Sólo en Chile, entre los años 1980 y 2016, se tiene registro de al menos 1000 personas fallecidas debido a este tipo de catástrofes, más de 28 personas al año ([41]), y en el terremoto del 27 de Febrero del año 2010 se registraron más de 2 millones de afectados ([19]). Para mitigar los efectos de estos daños, el factor crítico es la rapidez de la respuesta dentro de las primeras horas posteriores al evento, y la investigación y optimización de operaciones cobra vital importancia en esto.

Al producirse una catástrofe, las personas que quedan afectadas pueden estar alojando en refugios o albergues, o haber quedado varadas en zonas que se aislaron producto del desastre. Estas personas están en necesidad inminente de suministros (agua potable, comida, medicamentos, abrigo, entre otros), y estos deben ser distribuidos desde centros de acopio, donde se almacenan, organizan y envían. La distribución de suministros desde estos centros corresponde a un problema de elección de rutas para una flota de vehículos, sin embargo, se agrega la dificultad de que esto ocurre dentro de las primeras 72 horas después del incidente, y además, la cantidad de personas presentes en cada refugio, albergue o zona aislada es incierta, lo que no permite realizar un cálculo exacto de cuánto debe llevar y repartir cada vehículo. Debido a esto, la decisión de las rutas de cada vehículo, optimizando tanto el tiempo de entrega, ya que hay un factor de urgencia importante, como la entrega efectiva de suministros a las personas afectadas, resulta de gran importancia, ya que permite minimizar

la escasez de suministros en determinados refugios o albergues.

El problema de enrutamiento de vehículos es un problema combinatorial ampliamente estudiado en la literatura, y se han propuesto diversos algoritmos y técnicas diseñados para obtener las mejores soluciones posibles. En el siguiente trabajo se propone un modelo para el problema, combinando dos variantes del problema de enrutamiento de vehículos y se propone una solución a través del uso de metaheurísticas, particularmente el sistema de hormigas propuesto por Dorigo en [15].

El documento se divide en 5 secciones: en el capítulo 1 se describe el problema y su modelo matemático; en el capítulo 2 se expone una revisión del estado del arte; en el capítulo 3 se presenta el algoritmo de hormigas propuesto para esta variante del problema de enrutamiento de vehículos; en el capítulo 4 se describen y analizan los resultados obtenidos por la implementación del algoritmo y sus variedades y finalmente en el capítulo 5 se presentan las conclusiones del trabajo realizado.

Objetivos Generales

En este trabajo se propone un algoritmo basado en colonias de hormigas para el enrutamiento multi-depósito de vehículos bajo demanda estocástica, aplicado al contexto de respuesta a desastres naturales. Se construye además una implementación de este algoritmo, y, utilizando ésta, se realizarán experimentaciones con los parámetros variables del algoritmo, con el objetivo de mejorar las soluciones obtenidas. Luego, se compararán los resultados del programa con los presentes en la literatura.

En la revisión del estado del arte, se repasa el modelo clásico del problema de enrutamiento de vehículos, en conjunto con algunas de sus soluciones más conocidas. Además, se estudian algunas variantes del modelo, como por ejemplo, la variante multi-depósito y la variante de demanda incierta (o estocástica) del problema. Luego, se estudia el modelo clásico de la optimización por colonias de hormigas, y como dicho sistema ha contribuido a la solución del problema de enrutamiento de vehículos en su versión clásica.

En la evaluación experimental, se trabajará con los parámetros variables del algoritmo de hormigas, los cuales afectan de manera importante al comportamiento del algoritmo y por ende a la velocidad y la calidad de éste. Los resultados de dicha evaluación impactarán el algoritmo final para la comparación con casos de estudio existentes en la literatura.

Finalmente, en la fase de pruebas, el algoritmo será ejecutado usando instancias conocidas de la literatura, y posteriormente se realizará una comparación con los resultados obtenidos por otros algoritmos y soluciones bien conocidas.

El **objetivo principal** de este trabajo es proponer un algoritmo basado en colonias de hormigas para resolver el enrutamiento multi-depósito de vehículos, aplicado a distribución de suministros en desastres naturales.

A partir del objetivo principal se desprenden los siguientes **objetivos secundarios**

- Implementar el algoritmo propuesto
- Mejorar el algoritmo propuesto a través de experimentación usando los parámetros del sistema de hormigas
- Comparar el algoritmo con los presentes en la literatura usando instancias conocidas

Capítulo 1

Definición del Problema

El problema de enrutamiento de vehículos, de ahora en adelante **VRP** por sus siglas en inglés, se origina en la optimización de la distribución de productos y servicios y consta de lo siguiente: Dado un conjunto de puntos de demanda (también conocidos como clientes) en una zona, una bodega central y vehículos con cierta capacidad de distribución, se busca minimizar la distancia total recorrida por los vehículos, comenzando y finalizando cada una de estas rutas en el depósito central ([12]). En dicho trabajo, Dantzig y Ramser definen de manera formal el problema del siguiente modo:

- Dado un conjunto de n puntos de demanda, enumerados P_i ($i = 1, 2, \dots, n$), a los cuales se realizan entregas desde un punto P_0 , llamado punto terminal
- Una matriz de distancia $[D] = [d_{ij}]$ especificando las distancias $d_{ij} = d_{ji}$ entre cada par de puntos ($i, j = 1, 2, \dots, n$).
- Un vector de demandas $(Q) = (q_i)$ que especifica la cantidad demandada en cada punto P_i ($i = 1, 2, \dots, n$)
- La capacidad C del vehículo, con $C > \max\{q_i\}$
- $x_{ij} = x_{ji} = 1$ con el significado que indica que los puntos P_i y P_j están emparejados secuencialmente en la ruta y $x_{ij} = x_{ji} = 0$ indica que no lo están. Se agrega la condición

de que $\sum_{j=0}^n x_{ij} = 1$ para cada $i = 1, 2, \dots, n$

Del modelo anterior, se extrae la función objetivo del problema, la cual mide distancia recorrida por las rutas realizadas

$$\text{mín} \sum_{i=0}^n \sum_{j=0}^n d_{ij} \cdot x_{ij} \quad (1.1)$$

Se define una solución al problema como un conjunto de rutas que cubren los distintos puntos de demanda y cumplen correctamente con las restricciones definidas, y su costo como la distancia recorrida por estas rutas.

1.1. Variantes del VRP Relevantes para este Trabajo

A continuación se presentan algunas variables conocidas del VRP, y que son relevantes para esta investigación.

1.1.1. VRP Multi-Depósito

El VRP considera un solo depósito inicial desde el cual salen todos los vehículos y al cual tienen que regresar, pero si el problema cuenta con una cantidad mayor de depósitos, cada uno con una cantidad de vehículos determinada, y considerando que si un vehículo sale de un depósito en particular debe regresar al mismo, el problema se vuelve más complejo. A esta variedad se le conoce como MDVRP, o VRP Multi Depósito [10].

1.1.2. OVRP

Cuando los vehículos no deben regresar al depósito al finalizar el recorrido, el problema se abre. Esta nueva variante del VRP se llama Problema de Enrutamiento Abierto de Vehículos (**OVRP** por sus siglas en inglés [40]).

1.1.3. Demanda Estocástica en Emergencias

En [39], Qin et al. presentan una variación interesante del problema. Los autores proponen que para los casos de emergencia la demanda de cada punto de la zona es incierta, pues ocurre en un tiempo muy corto, además de que existen distintos puntos de distribución repartidos por todo el sector.

1.2. Modelo Matemático

1.2.1. Bases del modelo y notación

Para el caso de las emergencias, se considera la siguiente situación: al ocurrir el desastre natural, durante las primeras 48-72 horas no hay información concreta y verificable de la cantidad de personas afectadas, del volumen de las necesidades y de la ubicación de estas personas. Los suministros, por lo general, llegan de manera centralizada a la ciudad a través de ayuda coordinada u organismos gubernamentales, y se llevan a centros de almacenamiento (conocidos como centros de acopio) a lo largo de toda la zona afectada, desde los cuales se distribuyen a puntos de servicio: ya sean refugios, albergues o zonas aisladas. Dicha distribución de bienes se realiza a través de vehículos que pueden ser de origen gubernamental, propiedad del municipio, propiedad de las fuerzas armadas y bomberos, o propiedad de voluntarios. Debido a la falta de información disponible, la cantidad de personas en cada punto de demanda es incierta, por lo que se debe recurrir a métodos estadísticos de predicción para poder trabajar con estimaciones lo más certeras posibles.

Tomando todo eso en cuenta, el problema se modela como un VRP con las variantes siguientes: en vez de un solo punto central de distribución, existen diversos puntos repartidos a lo largo del mapa; la demanda de cada punto de servicio es incierta, y se ajusta a alguna distribución estadística conocida. Debido a la escasez de los recursos en este tipo de situaciones, distribuir suministros demás es un problema, ya que son suministros que debieron haberse destinado a otros puntos y rutas, y distribuir suministros de menos es un problema aún más grave, ya que puede traducirse en peores condiciones para los afectados o incluso pérdidas

humanas.

Para simplificar la representación y el problema, se considerará una capacidad homogénea para los vehículos, y que cada punto de demanda debe ser servido por solo un vehículo, supuestos que debido, a la naturaleza del contexto, se sabe que no se ajustan a la realidad, pero facilitan la proposición del algoritmo y representan una buena fuente para trabajos e investigaciones futuras respecto a este tema.

Para formular el modelo matemático, las notaciones a usar son las descritas en el Cuadro 1.1.

Cuadro 1.1: Notación y símbolos usados en el modelo

Símbolo	Descripción
R	Puntos correspondientes a refugios o albergues (puntos de demanda)
A	Centros de acopio y bodegas de suministros
P	$R \cup A$: todos los puntos relevantes de la zona
V	Conjunto de Vehículos
C	Capacidad máxima de cada vehículo
d_{ij}	Distancia entre los puntos i y j
ρ_r^-	Penalización por déficit de suministros en un punto r
ρ_r^+	Penalización por superávit de suministros en un punto r
$f_r(x)$	Función de probabilidad para la demanda en un punto r
θ_a	Cantidad máxima de vehículos disponibles en el centro de distribución a
e_r	Cantidad de suministros esperados (y entregados) en el punto r
k_r	Cantidad de suministros demandados reales en el punto r

La variable X_{ij}^v indica si un vehículo realizó un viaje determinado o no, y está representado con la siguiente expresión

$$X_{ij}^v = \begin{cases} 1 & \text{si el vehículo } v \text{ realizó el viaje entre las ciudades } i \text{ y } j \\ 0, & \text{de otro modo} \end{cases} \quad (1.2)$$

1.2.2. Modelo matemático

Como se mencionó en el punto anterior, el modelo deberá penalizar el error de predicción tanto en superávit como en déficit. Para ello, se considerará que la demanda en cada punto se ajusta a una función de probabilidad conocida, y se usará la esperanza de dicha distribución para calcular el valor esperado de suministros. Como se plantea en [39], en donde Qin et al. usan un modelo similar, la diferencia entre el valor demandado y el entregado queda representada así:

$$\Delta k_r^- = \text{máx}(0, k_r - e_r) \quad (1.3)$$

$$\Delta k_r^+ = \text{máx}(0, e_r - k_r) \quad (1.4)$$

Se pueden calcular también las probabilidades de que ocurra este déficit o superávit, usando la función de probabilidad $f_r(x)$. Estas probabilidades quedan descritas por las siguientes expresiones:

$$P_r^-(e_r) = P_r(k_r \leq z_r) = \int_0^{k_r} f_r(x) dx \quad (1.5)$$

$$P_r^+(e_r) = P_r(k_r \geq z_r) = \int_{k_r}^{+\infty} f_r(x) dx \quad (1.6)$$

A partir de esto se puede calcular el valor esperado del déficit/superávit para un punto r en particular, y queda definido del siguiente modo:

$$E(\Delta k_r^+) = \int_0^{e_r} (e_r - x) f_r(x) dx \quad (1.7)$$

$$E(\Delta k_r^-) = \int_{e_r}^{+\infty} (x - e_r) f_r(x) dx \quad (1.8)$$

Usando estos valores, la esperanza del déficit y del superávit para cada punto es calculable, y se integra a la función objetivo.

Se busca entonces minimizar el costo de todas las rutas sumadas, agregando penalizaciones para el déficit o superávit de recursos en cada punto. Por lo tanto, la función objetivo modelada para este trabajo queda entonces definida como:

$$\begin{aligned} \text{mín } \Psi = & \sum_{r \in R} (\rho_r^- \int_{e_r}^{+\infty} (x - e_r) f_r(x) dx + \rho_r^+ \int_0^{e_r} (e_r - x) f_r(x) dx) + \\ & \sum_{v \in V} \sum_{i \in P} \sum_{j \in P} d_{ij} \cdot X_{ij}^v \end{aligned} \quad (1.9)$$

El problema queda sujeto a las siguientes condiciones:

$$\sum_{j \in R} X_{ij}^v = \sum_{j \in R} X_{ji}^v \leq 1, \forall i \in A, \forall v \in V \quad (1.10)$$

$$\sum_{v \in V} \sum_{j \in R} X_{ij}^v \leq \theta_i, \forall i \in A \quad (1.11)$$

$$\sum_{i \in P} \sum_{j \in P} e_j \cdot X_{ij}^v \leq C, \forall v \in V \quad (1.12)$$

$$\sum_{i \in P} \sum_{v \in V} X_{ij}^v = 1, \forall j \in R \quad (1.13)$$

$$X_{ij}^v \in \{0, 1\}, \forall i, j \in P, v \in V \quad (1.14)$$

La restricción (1.10) indica que cada vehículo debe salir solo por un centro de acopio en cada ruta, y deben regresar a ese mismo centro al finalizar la jornada. La restricción (1.11) implica que no pueden salir desde determinado centro de acopio una cantidad mayor a la disponibilidad total de vehículos para ese centro. La condición (1.12) indica que la demanda total de una ruta no puede superar la capacidad total de los vehículos. La restricción (1.13) indica que cada refugio o albergue debe ser visitado una vez. Finalmente la expresión (1.14) indica la naturaleza de las variables.

Capítulo 2

Estado del Arte

A continuación se presenta una revisión de la literatura existente para el problema de enrutamiento de vehículos y sus variantes más conocidas, en conjunto con modelos de éste especializados en emergencias. Además, se presentan referencias en la literatura a los algoritmos de optimización por colonia de hormigas y métodos de mejoramiento de meta-heurísticas.

2.1. Solución al VRP

En [12], los autores proponen una solución al VRP a través de programación lineal, y a lo largo de los años se han propuesto numerosos algoritmos para obtener soluciones. En [9], Clarke y Wright proponen un algoritmo Greedy, y son los primeros en considerar la posibilidad de múltiples vehículos involucrados, demarcando oficialmente la existencia del problema como se le conoce ahora. En [28], publicado en 1992, Laporte realiza una revisión de los algoritmos postulados hasta el momento, trabajo que abre el paso a una gran explosión de soluciones y variantes del problema, producto de la disponibilidad y capacidad de la computación más moderna. Entre los años 1959 y 1992, prevalecieron algunos algoritmos exactos como lo son búsquedas a través de árboles generados relajando restricciones ([29], [8]), formulaciones en programación dinámica ([16]) y programación lineal entera ([1], [20], [17]). Para la comparación de los distintos algoritmos se usa, en la mayoría de los casos,

un set de instancias común: el propuesto por Christofides en 1977 ([6]), consistente en 14 instancias de carácter aleatorio, de las cuales las primeras 10 (etiquetadas *C1* a *C10*) tienen a los clientes distribuidos aleatoriamente por la zona, y las siguientes (*C11* a *C14*) tienen a los clientes agrupados. Los tamaños de estas instancias van entre los 50 clientes para la más pequeña (*C1*) a 199 para la más grande (*C10*). Además de las técnicas exactas mencionadas, se presentaron diversas técnicas heurísticas como soluciones para el problema, como por ejemplo el algoritmo de barrido ([24]), y el algoritmo greedy mencionado previamente ([9]). Sin embargo, los mejores resultados han sido los entregados por técnicas metaheurísticas. Según Blum y Roli en [2]:

Las metaheurísticas son conceptos de alto nivel para explorar espacios de búsqueda usando diferentes estrategias. Estas estrategias deben ser escogidas de una manera tal que se de un balance dinámico entre la explotación de la experiencia acumulada de la búsqueda (lo que comúnmente se llama *explotación*) y la exploración del espacio de búsqueda (lo que comúnmente se conoce como *diversificación*). Este balance es necesario por un lado para identificar rápidamente regiones del espacio con soluciones de alta calidad y por otro lado no perder mucho tiempo en regiones del espacio de búsqueda que ya fueron exploradas o que no proveen soluciones de calidad.

Estos algoritmos han gozado de una creciente popularidad en las últimas décadas, sobre todo para problemas que son combinatorialmente extensos y donde contar con estrategias que permitan viajar más rápidamente por el espacio de búsqueda es de vital importancia. Adicionalmente, debido al origen en el mundo real del VRP, durante los años han surgido distintas variantes interesantes del problema. A lo largo de esta sección se realizará una revisión de los métodos más efectivos de solución del VRP así como las variantes más relevantes en lo que a este trabajo concierne.

2.1.1. Algunos Algoritmos Propuestos

Tabu Search, de ahora en adelante TS, es un algoritmo de búsqueda metaheurística consistente en realizar búsqueda local, en principio, relajada, para luego ir agregando prohibiciones a medida que el algoritmo se estanca, de ahí el término *Tabú*. El algoritmo fue propuesto

por Glover en [25] y ha sido ampliamente utilizado en problemas de enrutamiento, como el VRP y el Traveling Salesman Problem ([22] y [30] respectivamente). Para el caso particular del VRP, TS ha logrado los mejores resultados de la literatura. En [22], los autores representan la ruta utilizando una lista de vértices (ciudades), e implementan una lista tabú de tamaño variable. Como variación a TS clásico, Gendreau, Hertz y Laporte utilizan dos técnicas: GENI, un procedimiento que aporta valor al proceso de inserción, introduciendo a las rutas sólo vértices vecinos de alguno de los vértices ya presentes en la lista, para luego aplicar un algoritmo de reoptimización de la ruta actual; además de US, una herramienta para post-optimización, la cual re-inserta todos los elementos de una solución usando GENI, para poder obtener el mejor orden posible de la ruta. Ambas técnicas fueron propuestas originalmente por los mismos autores en [21]. Las pruebas del algoritmo fueron realizadas con las instancias propuestas por Christofides, y los resultados observados fueron muy buenos, y son los más bajos encontrados en la literatura, con un tiempo de ejecución razonable (< 10 minutos para las instancias más pequeñas, < 100 minutos para las instancias más grandes).

También se han propuesto algoritmos genéticos (GA) para resolver el problema ([33], [38], entre otros). En [38], las soluciones se codifican como una secuencia de clientes sin delimitaciones de rutas, pudiendo interpretarse como la ruta que un solo vehículo realizaría para visitar a todos los clientes. Basándose en el principio de que *debe existir una secuencia óptima*, el algoritmo usa esta secuencia completa para luego, usando un procedimiento de separación, dividir la solución en rutas para cada vehículo. En cada iteración, los padres para la siguiente generación se obtienen a través de *selección por torneo binario*. Luego, la mutación se aplica en una tasa fija definida de antemano, y el procedimiento es un algoritmo de búsqueda local, convirtiendo a este algoritmo genético en híbrido. En esta búsqueda local los autores implementaron 9 movimientos disponibles, entre los cuales se observa por ejemplo la técnica *2-opt* para una sola ruta (intercambiando arcos dentro de un mismo viaje) y *2-opt* para varias rutas (intercambiando arcos de un viaje con los de otro). El primero de estos movimientos que represente una mejora es el elegido para mutar. El algoritmo es probado usando las instancias de Christofides ([6]), obteniéndose en el 71 % de las instancias el mejor resultado conocido (usualmente obtenidos por TS), pero en mejores tiempos de procesamiento (< 5 minutos para las instancias más pequeñas, < 60 minutos para las instancias más grandes).

Otra corriente cada vez más importante a la hora de buscar soluciones para el VRP es el sistema de hormigas, cuya descripción en particular se verá más adelante. Los primeros en proponer el uso del sistema de hormigas para el VRP fueron Bullnheimer, Hartl y Strauss en [3], obteniendo resultados comparables con los de TS.

2.2. Solución a las Variantes del VRP

2.2.1. VRP Multi-Depósito

En [10], Cordeau, Gendreau y Laporet proponen usar TS para resolver tres variantes del VRP: el MDVRP, el VRP Periódico (varias visitas a clientes a lo largo de una ventana de tiempo) y el Problema del Vendedor Viajero Periódico (TSP Periódico por sus siglas en inglés). Los autores generalizan el VRP de modo de poder resolver los tres problemas en un solo modelo, haciendo la analogía entre días y depósitos, términos que pueden ser intercambiables en este modelo. Así como en [22], de los mismos autores, el algoritmo usa la técnica de inserción optimizada GENI. El algoritmo propuesto es muy similar al de [22], guardando algunas diferencias fundamentales. La más grande de estas diferencias es la forma de generar una solución inicial, ya que repartir de manera arbitraria los puntos puede ser inviable debido a que la cantidad de días o de vehículos en cada depósito es limitada y específica para cada día/depósito. Lo que se hace en cambio es dividir el grupo por día o por cercanía a los depósitos, y se van agregando de manera aleatoria a las rutas para cada vehículo. Esto genera que sólo haya una ruta inviable al construirse la solución (la última). Otra diferencia fundamental es que en este caso la lista Tabú es de tamaño variable pero los elementos en ella tienen una *duración*, es decir, salen de la lista después de un número θ de operaciones. Para probar el algoritmo se utilizó un set de instancias del MDVRP de varias fuentes ([7], [23] y [4]). Los resultados obtenidos para las instancias son estrictamente mejores o iguales en el 74 % de los casos, comparándolos con algoritmos propuestos con anterioridad.

El año 2010, los autores proponen un GA para resolver esta variedad del problema. Al MDVRP inicial, se agrega una particularidad importante: El tiempo empleado en cada ruta, por lo general, se asume con la frase “a mayor distancia, mayor tiempo”, por lo que en

cuanto a la función objetivo se asume que minimizar la distancia es suficiente. Sin embargo, esto no es necesariamente verdadero, por ejemplo, un vehículo viajando varios kilómetros por una autopista puede tomar menos tiempo que un vehículo viajando unos pocos a través del tráfico en el medio de la ciudad. Debido a esto, los autores integran el tiempo a la función objetivo del problema. El modelo planteado indica una representación simple para cada cromosoma. La primera parte indica la cantidad de clientes a servir para cada vehículo del depósito, en orden para cada depósito. La segunda parte entonces indica las rutas de cada uno de estos vehículos. De este modo, la decodificación es directa, ya que con la primera sección se conocen las duraciones de las rutas de cada vehículo y esos números se usan para dividir la segunda parte. Para realizar el cruzamiento, el trabajo elige una operación *partial uniform and partial order* (PUPO), que indica cruzamiento uniforme para la primera parte del cromosoma (cambio en cada bit con una probabilidad determinada) y cruzamiento de orden para la segunda parte del cromosoma (intercambio de porciones completas de los padres). Para realizar las pruebas utilizaron el mismo set de instancias que Cordeau, Gendreau y Laporet en [10]. El algoritmo obtiene los mejores puntajes (conocidos con anterioridad) en el 66 % de las instancias, obtiene mejores resultados que los ya conocidos en 27 % y no logra el mejor resultado en el resto. Los tiempos de computación son muy buenos, demorándose en la instancia más grande menos de 4.5 minutos.

2.2.2. OVRP

Para la búsqueda de soluciones para el OVRP se han propuesto muchos algoritmos, la mayoría de carácter heurístico. En [31], los autores proponen el uso de un algoritmo natural basado en abejas llamado Honey Bees Mating Optimization, combinándolo con el algoritmo Expanding Neighborhood Search [32], donde la abeja madre se reproduce para obtener nuevas generaciones de abejas a través de vuelos (iteraciones de este proceso), generando así nuevas soluciones, con resultados muy cercanos a los de la literatura en general.

2.2.3. Problema de Enrutamiento de Vehículos para Emergencias

Sharif y Salari ([42]) presentan el problema como una variedad del OVRP, estableciendo que las rutas de entrega desde un depósito central pueden ser abiertas, es decir, sin terminar estrictamente en el depósito. Para resolver este problema, proponen el uso de un algoritmo GRASP (de **G**reedy **R**andomized **A**daptative **S**earch **P**rocedure). Los autores experimentaron con distintas técnicas de búsqueda local para la sección correspondiente de su algoritmo, como por ejemplo, *2-opt* o *string swap*. Dicha técnica obtuvo resultados competitivos versus algoritmos presentes en la literatura.

En [39], Qin et al. postulan un algoritmo genético para resolver el problema. Este algoritmo indica que para cada solución posible se penaliza la diferencia de demanda real versus la demanda estimada, e incluye tanto cruzamiento como mutación para las soluciones seleccionadas a través del método de la ruleta. Para cada centro de distribución se obtiene el conjunto de puntos de demanda a servir a partir de la distancia a dichos puntos, y a partir de esta división se resuelven múltiples VRP. El algoritmo propuesto obtiene muy buenos resultados para casos aleatorios propuestos por el equipo de investigación.

En [48], Yi y Kumar proponen también una solución usando un algoritmo de colonias de hormigas para resolver el problema, agregando la componente de traslado de personas heridas a servicios de emergencia.

2.3. Optimización por Colonias de Hormigas

La optimización por colonias de hormigas (**ACO** por sus siglas en inglés), propuesto por Dorigo en 1991-1992 ([14]), es un algoritmo metaheurístico y probabilístico proveniente de técnicas de **feedback positivo, dinámicas de colaboración y técnicas constructivas**. En ésta, diversos agentes, llamados hormigas, recorren el espacio de búsqueda para encontrar las soluciones óptimas, y, al igual que las hormigas reales, dejan un trazo de **feromonas** en el camino para que las hormigas que vienen tengan el conocimiento de cuales son los caminos más cortos a estas soluciones. Estas feromonas se evaporan con el tiempo, permitiendo de

esta manera al mismo tiempo la diversificación de las soluciones. Este algoritmo trabaja sobre grafos con peso no-negativos.

2.3.1. Modelo Clásico

La cantidad de feromonas presentes en un vertice del grafo es descrita a través de la ecuación (2.1), en la cual $\tau_{ij}(t)$ corresponde al valor de las feromonas en el camino entre i y j en un tiempo t , ρ es un coeficiente tal que $(1 - \rho)$ representa la evaporación en el camino entre un tiempo t y un tiempo $t + n$ y $\Delta\tau_{ij}^k$ corresponde a la cantidad total de feromonas depositadas entre los nodos i y j , entre t y $t + n$ por la hormiga k ([15]).

$$\tau_{ij}(t + n) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^m \Delta\tau_{ij}^k \quad (2.1)$$

El algoritmo comienza con soluciones vacías, y se van añadiendo nodos del grafo a través de una función de probabilidad de transición de un nodo i a un nodo j definida, clásicamente, de la manera siguiente

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{j \in v_k(t)} [\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}, & j \in v_k(t) \\ 0, & \text{de otro modo} \end{cases} \quad (2.2)$$

En la ecuación (2.2), τ_{ij} corresponde a las feromonas en el camino entre i y j , η_{ij} corresponde a una heurística local y v_k corresponde al conjunto de nodos no visitados por la hormiga k . α es un ponderador para el conocimiento empírico obtenido de las feromonas, y β le entrega ponderación a la heurística local escogida. Al hacer $\alpha = 0$, el algoritmo solo seguirá el camino más corto considerando la heurística, por lo que se convertiría en un algoritmo Greedy. La heurística local η_{ij} , por lo general, corresponde a la *visibilidad*, determinada por el recíproco de la distancia entre i y j .

Se define además una clase de agentes especiales llamadas *hormigas elitistas*, las cuales aumentan las feromonas al recorrer una solución. El número σ de hormigas elitistas a usar en

cada experimento está sujeto a sintonización, ya que el aumento de esta cantidad no implica necesariamente mejorar el resultado final: un exceso de hormigas elitistas podrían producir una convergencia prematura a soluciones no-óptimas ([3]).

2.3.2. ACO Aplicado al VRP

En [3], Bullnheimer et al. postulan un modelo de resolución del VRP usando un sistema de hormigas. En el, se usan tantas hormigas como puntos de demanda existan en el grafo, e intercambian sus lugares de inicio rotando su posición. Los autores proponen una actualización de la ecuación (2.1) para incluir la información de las hormigas elitistas, quedando entonces actualizada:

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \sum_{k=1}^m \Delta\tau_{ij}^k + \Sigma\Delta\tau_{ij}^* \quad (2.3)$$

Donde σ es el número escogido de hormigas elitistas y $\Delta\tau_{ij}^*$ es el aumento de feromonas producido por una hormiga elitista en un tiempo n entre los nodos i y j . Además de introducir elitismo, el modelo agrega un paso final a cada iteración, una técnica de búsqueda local. Luego de que una hormiga obtiene una solución factible, se chequea a través del procedimiento *2-opt* para comprobar que sea la solución más eficiente dentro de su vecindad de intercambios entre 2 arcos. De este modo, el algoritmo pasa a ser híbrido.

A lo largo de la primera parte de [3], los autores entregan analogías entre el VRP y el TSP, razón por la cual el sistema de hormigas que proponen para el VRP está basado en el sistema propuesto previamente para el TSP [13]. Sin embargo, también agregan que gracias a las particularidades del VRP se pueden introducir mejoras en el algoritmo. La primera particularidad es la existencia de un punto especial: el depósito central. En el VRP, no sólo importa la distancia relativa entre dos ciudades i y j , si no que también su distancia al depósito. Esta relación, llamada *ahorro*, se puede cuantificar con $\mu_{ij} = d_{i0} + d_{0j} - d_{ij}$. Un ahorro μ_{ij} indica que visitar al cliente i después del cliente j es una opción. Esto se traduce en una mejora a la

función de probabilidad (2.2), agregando la condición

$$p_{ij} \sim \mu_{ij}^\gamma \quad (2.4)$$

Donde γ regula el efecto de los ahorros en la probabilidad. Del mismo modo, es razonable intentar asegurar el mayor uso posible de la capacidad de cada vehículo. Siendo Q_i el total utilizado por un vehículo (incluyendo al cliente i), q_j la demanda del cliente j y Q la capacidad total del vehículo, entonces mientras más alto el valor de $\kappa_{ij} = \frac{Q_i + q_j}{Q}$, mejor la utilización del vehículo. Esto también se puede incluir como una condición para el sistema, agregando

$$p_{ij} \sim \kappa_{ij}^\lambda \quad (2.5)$$

Donde λ corresponde a la importancia relativa de κ_{ij} . Finalmente, las ecuaciones (2.4) y (2.5) se integran a la probabilidad total (descrita inicialmente en (2.2), del siguiente modo:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta \cdot [\mu_{ij}]^\gamma \cdot [\kappa_{ij}]^\lambda}{\sum_{j \in v_k(t)} [\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta \cdot [\mu_{ij}]^\gamma \cdot [\kappa_{ij}]^\lambda}, & j \in v_k(t) \\ 0, & \text{de otro modo} \end{cases} \quad (2.6)$$

Para probar estas mejoras, los autores usaron la instancia C1 del set de Christofides ([6]), probando con distintas configuraciones de α , β , γ y λ , y también haciendo uso o no de *2-opt* después de cada iteración. El resultado final fue que asignar igual importancia a los indicadores ($\alpha = \beta = \gamma = \lambda = 5$), e hibridizar el algoritmo entregó los mejores resultados. Además, experimentaron con el efecto de la existencia y número de hormigas elitistas en el algoritmo, llegando a la conclusión de que el número correcto de hormigas para esta versión del sistema era $\sigma = 30$. Agregando todas estas mejoras y configuraciones, se ejecutó el algoritmo usando todo el set de instancias. Los resultados obtenidos son competitivos con respecto al resto de los algoritmos, especialmente en el caso de las instancias que incluyen clientes agrupados (o en *clusters*), con diferencias con los óptimos conocidos de menos de 5 % para el 71 % de los casos. Los tiempos de ejecución no fueron reportados, pero se menciona que van desde menos de 1 minuto para las instancias más pequeñas y media hora para las instancias más grandes.

2.3.3. ACO aplicado a emergencias

El trabajo de Yi y Kumar, en [48], propone una variante interesante al VRP aplicado en el contexto de respuesta a emergencias. Los autores incluyen dos problemas en uno: primero, distribución de suministros, visto como un OVRP ya que los vehículos no necesitan regresar al depósito una vez que terminó su ruta; y segundo, transporte de personas heridas a servicios de emergencia. Las rutas que se van construyendo iterativamente, y al final de las iteraciones, se aplica el algoritmo Flujo Máximo Sucesivo (*SMF* por sus siglas en inglés), propuesto en [5], que se encarga de la redistribución de la ruta actual para intentar lograr el máximo flujo posible, intercambiando el flujo entre nodos vecinos usando operaciones *push* y redefiniendo la red usando *relabel*. A través de la experimentación, los autores notaron que usar el algoritmo de decisión greedy para la elección de los caminos más favorables convergió a soluciones no óptimas, debido al gran tamaño del espacio de búsqueda del problema. Debido a esto, Yi y Kumar plantean un depósito de feromonas llevado a cabo solamente por hormigas elitistas, las cuales resguardan el raking de σ mejores soluciones. Para realizar las pruebas de su algoritmo, los autores diseñaron 28 instancias aleatorias, de entre 20 y 80 puntos totales, con alrededor de un 30 % de puntos de servicio y hospitales, y el resto puntos de demanda. Se realizaron para cada una 30 iteraciones, y los resultados fueron casi tan buenos como los óptimos para dichas instancias, alejándose un máximo de 6.20 % de la mejor solución posible. Los tiempos de ejecución también fueron buenos, demorando un máximo de 40 segundos en las instancias más grandes.

2.4. Ajuste del algoritmo

A lo largo de los años, se han propuesto varias técnicas de ajuste para algoritmos metaheurísticos. En [35], los autores proponen un método llamado Estimación de Relevancia y Calibración de Valores en Algoritmos Evolutivos (*REVAC* por su sigla en inglés), en el cual se busca encontrar un vector de parámetros óptimo estimando la distribución de la función de utilidad del algoritmo en el dominio de cada parámetro. Comenzando con un vector base, el algoritmo estima una distribución y obtiene nuevos sujetos para la población base,

reemplazando uno a la vez. Luego del reemplazo, re-estima la distribución y obtiene nuevos vectores. Este método es simple, y entrega información valiosa respecto al comportamiento del algoritmo en relación a sus distintos parámetros, pero falla a la hora de entregar información entre la relación entre los parámetros en si. En [34], los autores recurren a un modelo estadístico para optimizar los parámetros de su ACO, dejando parámetros fijos y variando uno a la vez para estimar el efecto y los valores óptimos para cada valor.

Otras alternativas para realizar este procedimiento son las hiper-heurísticas. Introducidas por primera vez en [11], las hiper-heurísticas se definen como algoritmos que operan por sobre las meta-heurísticas regulares, con un dominio de información mucho más acotado. El objetivo es ajustar los parámetros de una meta-heurística objetivo, pudiendo de este modo construir un procedimiento semi-agnóstico del dominio del problema. Gracias a esta característica se puede aplicar una misma hiper-heurística a clases completas de problemas. En [27], Jacobsen-Grocott et al. proponen un meta algoritmo (hiper-heurística) para el VRP con ventanas de tiempo. Este algoritmo es de tipo genético, y se construye para mejorar el resultado final objetivo, entregando resultados competitivos.

Capítulo 3

Propuesta

Para encontrar soluciones a este problema, se decide diseñar un algoritmo basado en la optimización por colonias de hormigas, debido a que la combinación entre el uso del aprendizaje proveniente del uso de matrices de feromona y la aleatoriedad intrínseca del modelo entregan un balance muy valioso a la hora de diversificar y explotar espacios de soluciones. Utilizando la existencia de múltiples depósitos, el problema general se divide en varios sub-problemas de menor tamaño, uno para cada depósito, asignando entonces un conjunto de puntos de demanda a los cuales debe servir. En el cuadro 3.1 se describen los símbolos a usar en el algoritmo.

Cuadro 3.1: Notación y símbolos usados en el algoritmo

Símbolo	Descripción
η_{ij}	Visibilidad desde i a j
D_a	Conjunto de puntos que el depósito a debe servir
$Q_T(a)$	Cantidad total de demanda para los puntos asignados al depósito a
ν	Conjunto de puntos disponibles en un momento determinado

3.1. Representación

Para representar una solución al problema, se utiliza una lista donde cada elemento corresponde al conjunto de rutas realizadas por vehículos asignados a un determinado depósito. Para representar la ruta de cada vehículo se usa una lista de tuplas, donde cada tupla contiene dos elementos: el punto al que el vehículo se moverá, y la cantidad de suministros que va a entregar en ese punto. Cada una de estas rutas comienza y finaliza con el depósito en cuestión. En estos pasos, el segundo elemento de la tupla es un cero, debido a que no se entregan suministros. De esta manera entonces, una solución completa es representada por una lista compuesta por listas de rutas. En la figura 3.1 se puede observar un esquema de la representación propuesta.

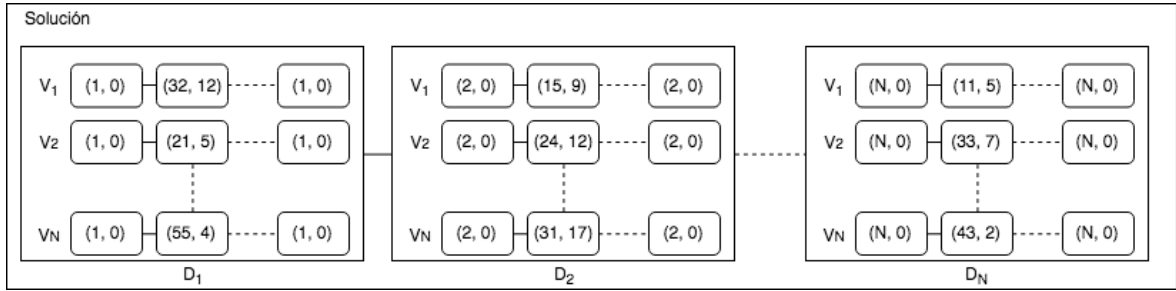


Figura 3.1: Representación de una solución. Cada ruta es representada por una lista de tuplas punto - cantidad distribuida, cada depósito es representado por una lista de rutas, una para cada vehículo, y finalmente la solución completa es representada por una lista de cada una de las soluciones para sus depósitos.

Para ejemplificar el uso de esta representación, se utiliza como ejemplo las rutas propuestas en la figura 3.2, correspondientes a la instancia de prueba utilizada por Qin et al. en [39].

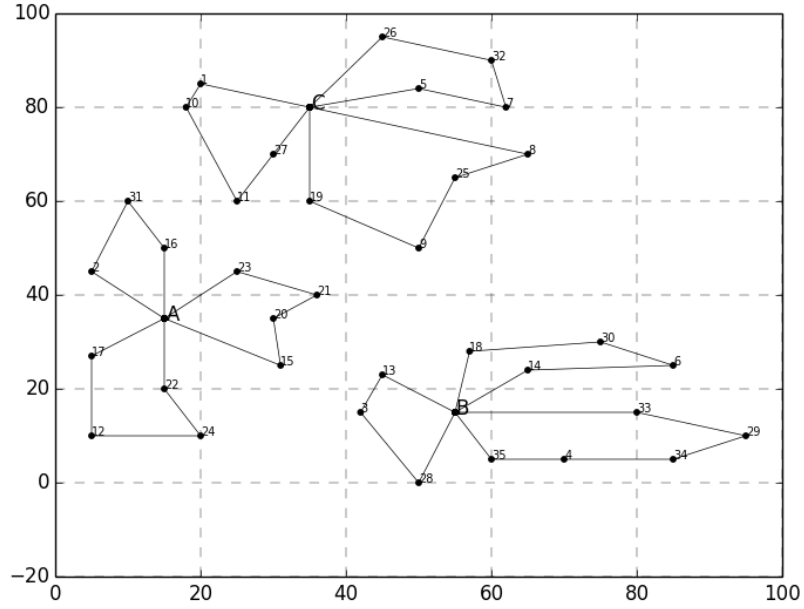


Figura 3.2: Ejemplo de solución propuesta

3.2. Función de evaluación

La función de evaluación con la que trabajan las hormigas para decidir la calidad de las soluciones es la planteada en la expresión 1.9, reiterada a continuación:

$$\begin{aligned} \text{mín } \Psi = & \sum_{r \in R} (\rho_r^- \int_{e_r}^{+\infty} (x - e_r) f_r(x) dx + \rho_r^+ \int_0^{e_r} (e_r - x) f_r(x) dx) + \\ & \sum_{v \in V} \sum_{i \in P} \sum_{j \in P} d_{ij} \cdot X_{ij}^v \end{aligned} \quad (3.1)$$

En esta función existen dos componentes claros: Dos integrales que permiten calcular la penalización de entregar suministros con un déficit o un superávit, y la sumatoria clásica del problema de ruteo de vehículos, la cual considera la distancia recorrida como el costo de la solución.

3.3. Estructura

El algoritmo se estructura como se muestra en el pseudo-código 1. En cada iteración, cada hormiga realiza un recorrido completo para uno de los depósitos. Así, cada una obtiene su propia ruta que cubre cada uno de los puntos asignados a su correspondiente depósito. Luego, se realiza un procedimiento de búsqueda local entre las rutas obtenidas por las hormigas, para luego dividir la ruta final de acuerdo a la cantidad de vehículos y el cumplimiento de las restricciones de capacidad y demanda. A continuación, se detalla cada uno de los componentes del algoritmo.

3.3.1. Inicialización

El sistema se inicializa con la matriz de feromonas con todos los valores iguales para cada camino entre puntos (ya sean refugios o centros de acopio). Esto permite que al comenzar las iteraciones todos los puntos tengan las mismas probabilidades de ser visitados, sin tener caminos favoritos predeterminados.

3.3.2. Asignación de puntos de demanda a depósitos

Se busca dividir el problema en múltiples VRP en lugar de resolver para todos los puntos al mismo tiempo, por lo que se debe separar el conjunto de puntos de demanda según su distancia a los centros de acopio. Esto sigue el paradigma de *Dividir y Conquistar*, con el que se busca granular el problema para simplificar el diseño del algoritmo. Cada punto de demanda será asignado al depósito que tenga más cercano, exceptuando aquellos casos donde se produzca un empate. Un empate se define como un caso en el que un punto de demanda cumpla la condición 3.2, es decir, donde la distancia entre los dos depósitos más cercanos (A y B) estén comparablemente cerca del punto. Para esto se usa el parámetro a_{dist} . En caso de empate, se consideran los a_{cut} depósitos más cercanos al punto de demanda y se selecciona uno al azar.

$$|D_{pA} - D_{pB}| \leq a_{dist} \quad (3.2)$$

```

inicialización;
best_sol = Solución Vacía;
foreach iteracion do
    sol = Solución Vacía;
    foreach depot  $\in A$  do
        H = crear m hormigas;
        foreach hormiga  $\in H$  do
             $v = D_{depot}$ ;
            punto_inicial = rand( $v$ );
            remover punto_inicial de  $v$ ;
            hormiga.comenzar_nueva_ruta;
            hormiga.ruta.agregar punto_inicial;
            while  $v$  no vacío do
                probabilidades = hormiga.calcular_probabilidades( $v$ );
                if probabilidades vacío then
                    hormiga.cerrar_ruta;
                    hormiga.comenzar_nueva_ruta;
                else
                    cd_selección = hormiga.seleccionar_ciudad( $v$ , probabilidades);
                    hormiga.ruta.agregar selección;
                    remover selección de  $v$ ;
                end
            end
        end
        sol[depot].agregar mejores(H, CROSS-exchange(H), Or-opt(H));
        if best_sol < sol then
            | best_sol = sol
        end
    end
end

```

Algorithm 1: Algoritmo propuesto

El proceso de división del problema se repite cada a_{iter} iteraciones ejecutadas, de manera de otorgar a las hormigas la posibilidad de encontrar rutas nuevas en base a nuevas combinaciones de nodos.

3.3.3. Estructura de cada iteración

Desplazamiento y construcción de rutas

En cada iteración, se asigna un número fijo m de hormigas a cada depósito, cada una de las cuales recorrerá el conjunto completo de puntos de demanda asignados al depósito. Para cada hormiga, la iteración comienza con una ruta que inicia en un punto determinado al azar, y se inicializa una lista llamada **puntos posibles** la cual contiene todos los puntos de demanda asignados al depósito menos aquel que fue escogido para comenzar. Las hormigas entonces realizan sus movimientos decidiendo a que punto desplazarse utilizando la función de transición descrita en la expresión 3.3. Los factores que forman parte de dicha decisión son: τ_{ij} , factor que representa la cantidad de feromonas en el camino entre i y j ; η_{ij} corresponde a la visibilidad, es decir, el recíproco de la distancia entre i y j ; μ_{ij} , indicador del ahorro obtenido al visitar al punto j después del punto i ; y finalmente, Z_{ij} , que indica el uso de la capacidad total del vehículo una vez visitado el punto j luego del punto i . Estos indicadores están ponderados utilizando los parámetros α , β , γ y λ , respectivamente.

$$p_{ij}^v = \begin{cases} \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta \cdot [\mu_{ij}]^\gamma \cdot [Z_{ij}]^\lambda}{\sum_{j \in v} [\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta \cdot [\mu_{ij}]^\gamma \cdot [Z_{ij}]^\lambda}, & j \in v \\ 0, & \text{de otro modo} \end{cases} \quad (3.3)$$

Las hormigas visitan cada uno de los puntos disponibles en la lista **puntos posibles**, y cada vez que realizan un desplazamiento eliminan dicho elemento de las posibilidades. Una vez que las listas de **puntos posibles** de todas las hormigas del depósito están vacías, se procede a realizar la búsqueda local.

Búsqueda Local

Al final de la etapa de desplazamiento, cada hormiga representa entonces una solución factible para su depósito. Para mejorar las soluciones y entregar variedad al vecindario, se aplican tres operadores de búsqueda local: *CROSS-Exchange*, *Or-opt* y *Relocate*.

Para cada depósito se toma la hormiga con la ruta menos costosa obtenida, y se aplican los operadores, descritos a continuación.

CROSS-exchange – propuesto en [46], trabaja con un par de rutas intercambiando segmentos completos de igual tamaño entre ellas, pero manteniendo la posición de dichos segmentos. De este modo, es mucho más probable que las restricciones de tiempo se sigan cumpliendo por lo que la solución resultante sigue siendo válida. Para el caso de este algoritmo, se aplica el operador para cada par de rutas disponible, usando segmentos de todos los tamaños posibles (tamaño 1 hasta $l_s - 1$, donde l_s es el tamaño de la ruta más pequeña entre las dos), y realizando el cambio en todos los posibles puntos de inicio y destino, cubriendo de este modo todas las combinaciones posibles para la operación. La figura 3.3 muestra el funcionamiento del operador.

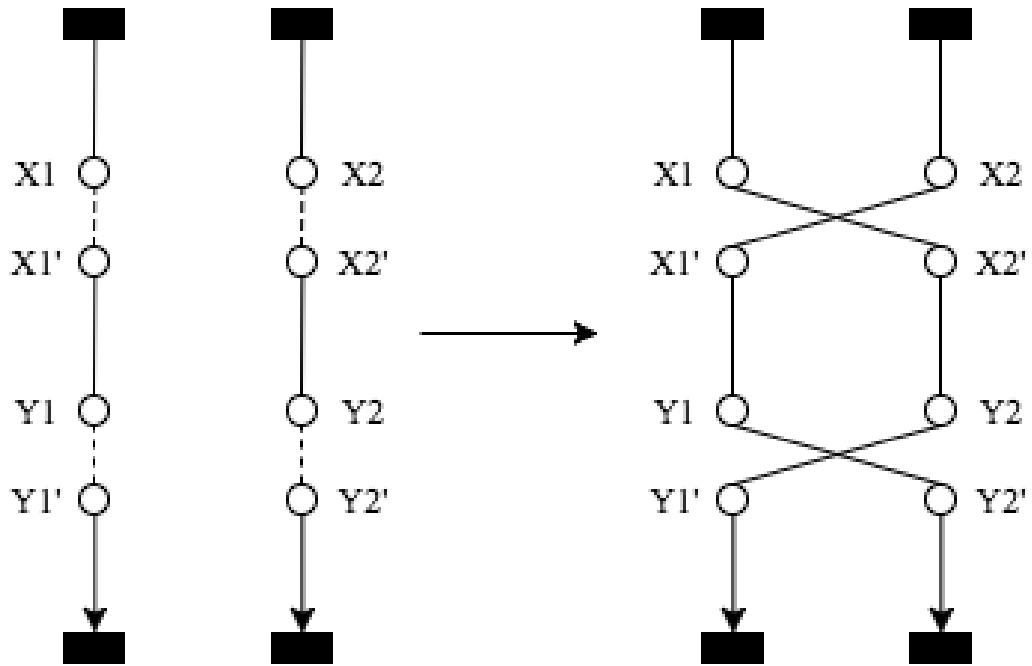


Figura 3.3: CROSS Exchange en acción.

Or-opt – es un caso especial de CROSS exchange, propuesto por primera vez en [37]. Esta técnica de búsqueda local recorta parte de una ruta y la inserta completamente, manteniendo la posición temporal de los puntos, en otra ruta. Para el caso de este algoritmo, se aplica el operador para cada par de rutas disponible, usando segmentos de todos los tamaños posibles (tamaño 1 hasta $l_s - 1$, donde l_s es el tamaño de la ruta más pequeña entre las dos), y realizando el cambio en todos los posibles puntos de inicio y destino, cubriendo de este modo todas las combinaciones posibles para la operación. La figura 3.4 muestra el operador en funcionamiento.

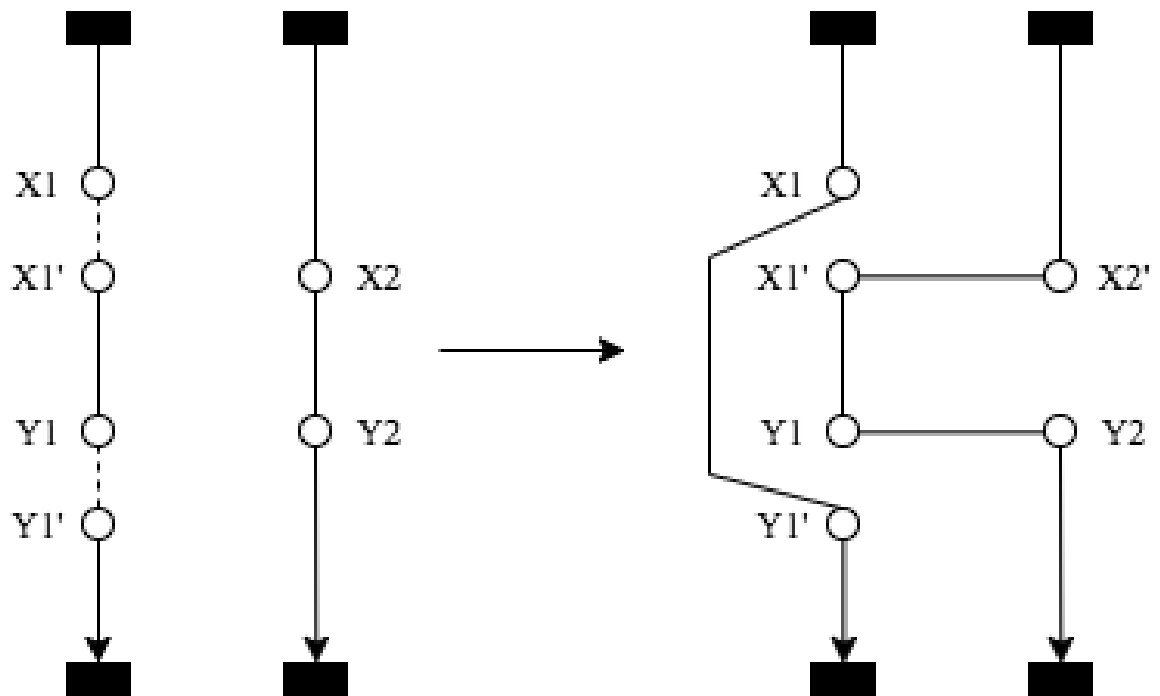


Figura 3.4: Or-Opt en acción.

Relocate – es un operador que toma un elemento y lo traslada cierto número de posiciones hacia adelante o hacia atrás dentro de una ruta, sin alterar el orden del resto de los elementos de la ruta. Para este algoritmo, se prueba mover cada uno de los elementos de cada una de las rutas tantas posiciones como queden disponibles hacia atrás y hacia adelante. La figura 3.5 muestra el operador y su funcionamiento.

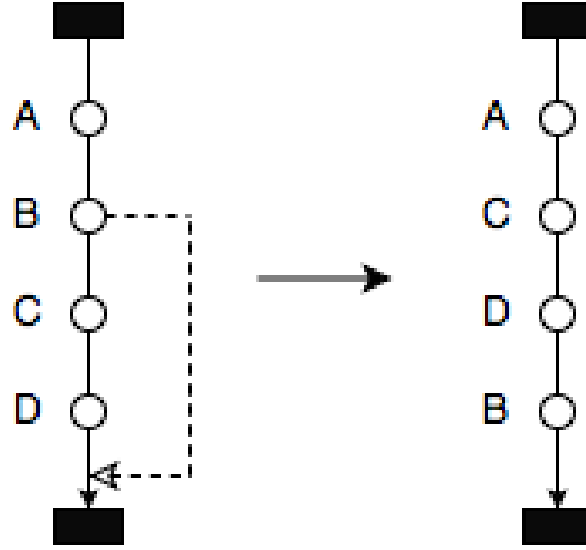


Figura 3.5: Relocate en acción.

Al aplicar dichos operadores, se expande el vecindario de soluciones posibles. Una vez obtenido el vecindario completo, se descartan todas las soluciones que no cumplan con las restricciones del problema, de modo de contar solamente con un grupo de soluciones viables, y se calculan los costos de cada una. El conjunto de rutas que entregue el menor costo es seleccionada como la solución para el depósito en cuestión, utilizando la fórmula (1.9). La elección de una estrategia de mejor-mejora versus una estrategia primera-mejora se discute en la sección 4.3.1.

Actualización de la matriz de feromonas

Al finalizar el procedimiento, se actualiza la matriz de feromonas con la expresión 3.4, para todos los arcos que fueron usados en los recorridos de los vehículos en la solución seleccionada luego de la búsqueda local. En dicha expresión, τ_{ij}^{nueva} representa el nuevo valor de las feromonas en la posición (i, j) de la matriz, y $\Delta\tau_{ij}^v$ para cada hormiga v equivale al recíproco del valor obtenido con la función de costo para dicha hormiga.

$$\tau_{ij}^{nueva} = \rho \cdot \tau_{ij}^{anterior} + \sum_{v=1}^m \Delta\tau_{ij}^v \quad (3.4)$$

Además, se considera la propuesta de Stützle y Hoos en [44] de utilizar un límite máximo y un límite mínimo para las feromonas, definiendo los límites como los siguientes:

$$\tau_{ij}^{\max} = \frac{1}{1 - \rho} \cdot \frac{1}{f(opt)} \quad (3.5)$$

donde $f(opt)$ corresponde al costo total de la mejor solución encontrada hasta ahora, por lo que el valor es actualizado cada vez que se encuentra una mejor solución. Para el caso de $\tau_{ij}^{\min}$, se define mediante la siguiente expresión:

$$\tau_{ij}^{\min} = (1 - \tau_{ij}^{\max}) \cdot (1 - \rho) \quad (3.6)$$

De este modo entonces, al actualizar la feromona se acota a estos límites definidos. La incorporación de ésta técnica se contrasta con la actualización de feromonas clásica en la sección 5.2, de modo de entender el impacto que tiene en la calidad de los resultados del algoritmo.

3.3.4. Criterios de detención

Debido a la naturaleza del algoritmo, puede ser ejecutado de manera indefinida, por lo que se definen criterios de detención basados en tiempo o en número de iteraciones sin mejora. Se especifican más detalles al respecto en el capítulo 4.

3.3.5. Resumen

El algoritmo propuesto trabaja dividiendo el problema global en varios sub-problemas a abordar, para luego realizar iteraciones en las cuales las hormigas encuentran rutas usando todos los puntos del depósito al que están asignadas. Los parámetros que configuran este algoritmo se detallan en la tabla 3.2. Los parámetros α , β , γ , λ se utilizan en el depósito de feromonas, mientras que $iter$, $dist$ y $corte$ se usan para determinar cambios en la asignación de puntos de demanda a depósitos. m corresponde a la cantidad de hormigas a usar y ρ corresponde a la persistencia de las feromonas tras cada iteración.

Cuadro 3.2: Parámetros del algoritmo

Parámetro	Descripción
m	Cantidad de Hormigas por depósito
ρ	Persistencia de las feromonas tras cada iteración
α	Ponderador de las feromonas
β	Ponderador de la visibilidad
γ	Ponderador del ahorro
λ	Ponderador de la utilización
a_{iter}	Número de iteraciones tras el cual reasignar depósitos
a_{dist}	Distancia umbral para determinar empate entre depósitos
a_{corte}	Número de depósitos entre los que se realizará la decisión aleatoria en caso de ocurrir empate

Capítulo 4

Experimentos y Resultados

En este capítulo se presentan los experimentos y resultados orientados a validar el algoritmo propuesto para el problema de enrutamiento de vehículos estudiado. A continuación se describe el ambiente de experimentación diseñado, estableciendo en detalle los objetivos del análisis. Luego, se presentan los resultados y las conclusiones extraíbles de cada uno de los análisis realizados.

4.1. Experimentación

En este capítulo se evalúa el desempeño del algoritmo propuesto. Para esto se han diseñado tres conjuntos de experimentos cuyos objetivos se detallan a continuación.

1. Comprobar la efectividad de dos posibles estrategias de búsqueda local: mejor mejora y primera mejora. La estrategia de mejor mejora busca computar el vecindario completo de soluciones utilizando todos los operadores de búsqueda local, para todas las opciones de su dominio. Una vez computado el vecindario, se toma la solución con el menor valor. Para el caso de primera mejora en cambio, se computan las soluciones vecinas y, cada vez que se calcula una nueva, se verifica si tiene un valor menor al de la

solución original. De ser el caso, se toma esa solución sin computar el resto del vecindario. En ambos casos se usan todos los operadores disponibles: CROSS-Exchange, Or-Opt y Relocate.

2. Evaluar que modelo de MaxMin satisface de mejor manera el problema: MaxMin clásico, cambiando los valores de $\tau_{ij}^{\min}$ usando la estrategia propuesta en [44] y $\tau_{ij}^{\max}$; MaxMin fijo, utilizando $\tau_{ij}^{\min}$ y $\tau_{ij}^{\max}$, variando los valores fijos dependiendo de la instancia utilizando para calcularlos los mejores resultados encontrados en la literatura; y una variante de MaxMin fijo, pero en lugar de usar límites distintos para cada instancia, se utiliza $\tau_{ij}^{\max} = 4$ y $\tau_{ij}^{\min} = 0$.
3. Evaluar una alternativa menos costosa de búsqueda local que seleccione en cada paso qué operador, de los tres disponibles, utilizar. Así, en cada paso se elige un único operador: Or-Opt, CROSS-Exchange o Relocate. Esta alternativa es menos costosa y fomenta la exploración del vecindario.
4. Comparar con el estado del arte. Se consideran acercamientos basados en búsqueda local propuestos en la literatura que resuelven las mismas instancias estudiadas en este trabajo.

4.2. Instancias de Prueba

Se escoge un marco de referencia correspondiente a las instancias propuestas por Christofides en [6]. Estas instancias corresponden al marco de referencia clásico en la literatura del problema de enrutamiento de vehículos. En el cuadro 4.1 se presenta una vista general de las instancias a considerar. En el cuadro 4.1, I corresponde al identificador de la instancia, n es el número de puntos de demanda y d es el número de depósitos. Las columnas C^* y T corresponden al mejor resultado encontrado en la literatura y su tiempo de ejecución. Para todos estos casos de prueba, los mejores resultados fueron obtenidos por el algoritmo propuesto por Cordeau, Gendreau y Laporte en [22].

Cuadro 4.1: Instancias de Christofides

I	n	d	C^*	T
C1	50	4	576.9	194.4
C2	50	4	473.9	207.6
C3	75	5	645.5	339.6
C4	100	2	1006.5	467.4
C5	100	2	753.3	492.6
C6	100	3	877.8	459
C7	100	4	892	462.2
C8	249	2	4482.4	1525.8
C9	249	3	3920.9	419.4
C10	249	4	3714.7	984.0
C11	249	5	3580.4	1040
C12	80	2	1319.0	334
C15	160	4	2505.4	844
C18	240	6	3702.9	1455

4.2.1. Equipamiento técnico

Los resultados de las pruebas fueron obtenidos utilizando un equipo con las siguientes características técnicas: Amazon Linux con 16 GB de memoria RAM (T2.xlarge en AWS EC2). El programa fue construido utilizando C++14.

4.2.2. Sintonización de Parámetros

Para sintonizar los parámetros del algoritmo, se hace uso de ParamILS. Propuesto en [26], ParamILS es un framework de sintonización automático que utiliza búsqueda local iterada para encontrar la mejor configuración de parámetros para un algoritmo utilizando una clase de problemas de la misma categoría.

Para aplicar ParamILS en el algoritmo propuesto, se definen los dominios deseados de cada

Cuadro 4.2: Parámetros sintonizados por ParamILS. Para cada parámetros se presenta su conjunto de valores posibles y valores iniciales.

Parámetro	Valores Disponibles	Valor Inicial
ρ	0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 0.99	0.99
α	1, 2, 3, 4, 5	1
β	1, 2, 3, 4, 5	1
γ	0, 1, 2, 3, 4, 5	1
λ	0, 1, 2, 3, 4, 5	1
m	5, 10, 15, 20	5
a_{iter}	10, 100, 200, 1000	1000
a_{dist}	0.1, 0.2, 0.3, 0.4	0.1
a_{corte}	2, 3	3

uno de los parámetros de la tabla, un conjunto finito de valores posibles para cada parámetro y un valor inicial para el proceso de sintonización. Dichos valores se presentan en la tabla 4.2. El parámetro ρ indica la persistencia de las feromonas iteración tras iteración, y se eligió un conjunto de valores desde 0.6 a 0.99. Los casos $\rho < 0.6$ presentan mucha volatilidad a los valores de las feromonas. Para α y β , los parámetros de ponderación clásicos de los algoritmos de optimización por colonias de hormigas, se eligieron los valores de 1 a 5, y para γ y λ , valores de ponderación para el ahorro de viajar de un punto a otro y la capacidad disponible del vehículo respectivamente, se eligieron valores entre 0 y 5. Para la cantidad de hormigas se eligieron valores estándar de la literatura entre 5 y 20, y para los parámetros que controlan la asignación GRASP se eligieron valores que cubrieran gran parte del dominio y fueran representativos. Estos parámetros son a_{iter} , que controla la cantidad de iteraciones sin mejora después de las cuales se realiza la re-asignación de puntos, a_{dist} , la distancia porcentual entre los dos depósitos más cercanos que sirve como umbral para decidir si asignar al azar, y a_{corte} , el número de depósitos entre los que se re-asignará al azar el punto de demanda en caso de que se cumpla el umbral a_{dist} entre los dos depósitos más cercanos.

Como conjunto de entrenamiento para el proceso de sintonización se escogen las primeras

Cuadro 4.3: Configuración de parámetros entregada por ParamILS

Parámetro	Valor
ρ	0.99
α	5
β	2
γ	1
λ	2
m	15
a_{iter}	10
a_{dist}	0.4
a_{corte}	2

cuatro instancias de Christofides [6]: C1, C2, C3 y C4 en la tabla 4.1. Este conjunto de instancias provee variedad respecto a número de clientes y disposición de puntos de demanda. Para el proceso de sintonización se utiliza como medida de calidad de cada ejecución lo que se conoce como distancia al óptimo. Esta distancia se calcula usando la expresión 4.1, en la cual C es el costo obtenido luego de la correspondiente ejecución del algoritmo y C^* corresponde al mejor resultado encontrado en la literatura para dicha instancia de prueba.

$$C' = \frac{C}{C^*} \quad (4.1)$$

Luego de ejecutar ParamILS para la configuración descrita, los resultados obtenidos indican que la configuración óptima del algoritmo es la expuesta en el cuadro 4.3. Los resultados obtenidos se acercan a lo esperable para los parámetros de ponderación, feromonas y hormigas, ya que son valores similares a los encontrados en la literatura. Para el caso de la asignación GRASP de puntos de demanda, los resultados de los parámetros indican que re-asignar rápidamente, con $a_{iter} = 10$, el valor más bajo, y utilizar umbral alto de distancia con $a_{corte} = 0.4$, el valor más alto, tiene mejores resultados que confiar más en la distancia explícita, validando entonces la utilización del método.

4.2.3. Detalles de la ejecución

Cada instancia fue ejecutada utilizando un conjunto de 20 semillas distintas. Se considera como criterio de detención un tiempo máximo de ejecución medido en segundos. Para las instancias C1-C7 y C12-C18 se consideraron 600 segundos y para las instancias C8-C11 se consideraron 1200 segundos.

4.3. Resultados

En esta sección se presentan los resultados obtenidos a partir del proceso de experimentación propuesto. Para cada uno de los análisis se presentan los resultados de calidad y tiempo de cómputo de cada uno de los esquemas analizados usando boxplots. Un boxplot es un método para graficar grupos de datos numéricos usando cuartiles. Para cada grupo de datos, el boxplot muestra, de izquierda a derecha, el valor mínimo de la muestra, la mediana, los cuartiles y el valor máximo. Observando los boxplots se puede ver si las distribuciones son simétricas, si existen outliers en la muestra, y la distribución de los datos a lo largo de la escala de cuartiles. Cada uno de los boxplots fue construido en base a los resultados de 20 ejecuciones independientes del algoritmo indicado en cada caso.

4.3.1. Esquema de Búsqueda Local

En esta sección se presentan los resultados comparativos de los dos esquemas de búsqueda local estudiados. Los esquemas utilizados fueron mejor mejora versus primera mejora. Mejor mejora consistente en computar el vecindario completo de soluciones de búsqueda local utilizando todos los operadores: CROSS-Exchange, que se encarga de intercambiar tramos de las rutas entre sí, Or-Opt, que funciona desviando una ruta a un tramo originalmente perteneciente a otra, y Relocate, que traslada puntos dentro de la misma ruta a otras posiciones. Primera mejora, consistente en encontrar la primera solución del vecindario que presente alguna mejora en lugar de computar todo el vecindario considerando los tres operadores de

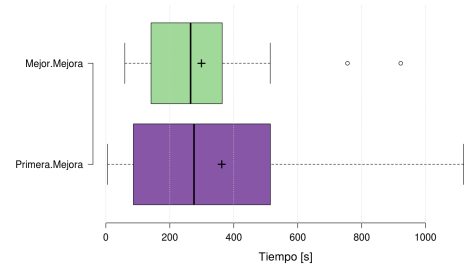
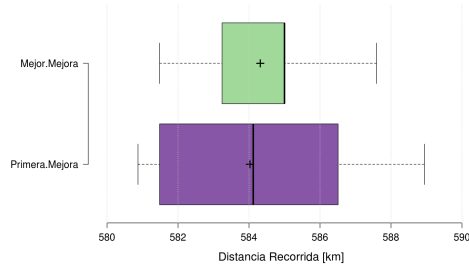
búsqueda local implementados. Las figuras 4.1 a 4.4 muestran para cada una de las instancias estudiadas, la calidad de las soluciones encontradas (en Km recorridos) y el tiempo de cómputo de los algoritmos comparados (en segundos). En color verde claro se muestran los resultados de la versión mejor mejora y en morado la versión primera mejora.

Caso

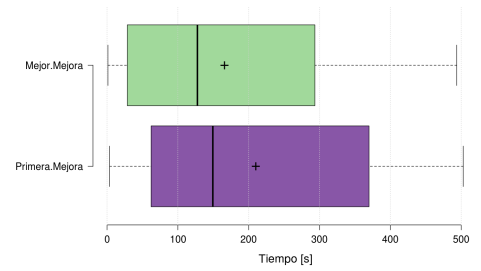
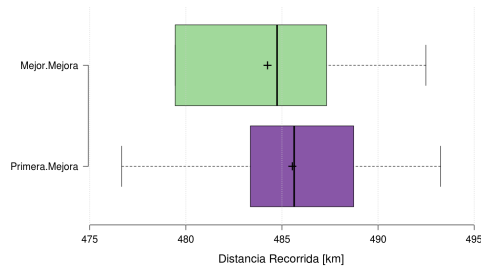
Distancia [km]

Tiempo [s]

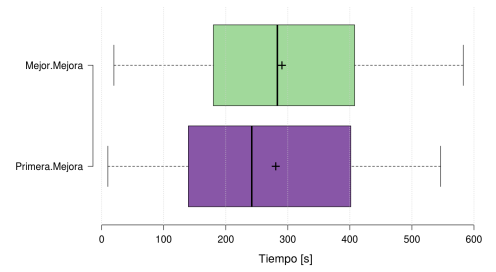
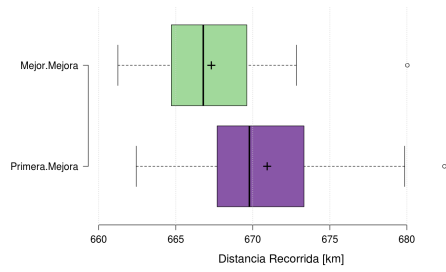
C1



C2



C3



C4

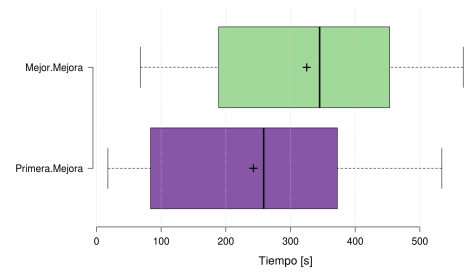
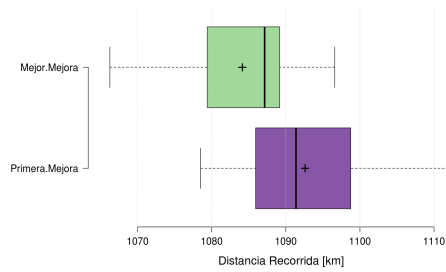


Figura 4.1: Comparación Búsqueda Local: Primera mejora versus Mejor Mejora – Instancias de Christofides: C1 - C4

La figura 4.1 muestra la comparación para las instancias C1 a C4. En este caso es posible observar que los mejores resultados se obtienen, en la mayoría de los casos, usando la versión mejor mejora.

Respecto a los tiempos, es posible observar que para C1 y C2 los tiempos de la versión mejor mejora son superiores, pero esta situación cambia para las instancias C3 y C4. Esto se produce debido a que, al aumentar el tamaño de la instancia de 50 a 75 y 100 puntos de demanda, la estrategia de mejor mejora multiplica su tiempo usado en búsqueda local ya que la cantidad de combinaciones para el uso de los operadores aumenta de manera significativa.

La figura 4.2 muestra la comparación para las instancias C5 a C8. Para este conjunto de instancias es posible observar que los mejores resultados se obtienen siempre usando la versión mejor mejora. Los tiempos de cómputo lucen bastante similares para C7 y C8, pero presentan medias claramente inferiores en la versión mejor mejora para las instancias C5 y C6.

Mejor mejora en estos casos tiene la ventaja de que cubre el vecindario completo aplicando CROSS-Exchange, Or-Opt y Relocate, por lo que al computar todas las alternativas a partir de una solución original toma una decisión más informada para dar el siguiente paso. Primera mejora, por otro lado, puede demorarse más en llegar a soluciones de esa calidad debido a que se mueve, posiblemente, basado en mejoras más pequeñas.

La figura 4.3 muestra la comparación para las instancias C9 a C12. Con respecto al tiempo de ejecución, entre C9 y C11 se observa que los resultados son parejos, con la media de primera mejora siendo levemente mejor que la de mejor mejora. En el caso de C12 esta diferencia es más notoria con primera mejora obteniendo claramente mejores resultados.

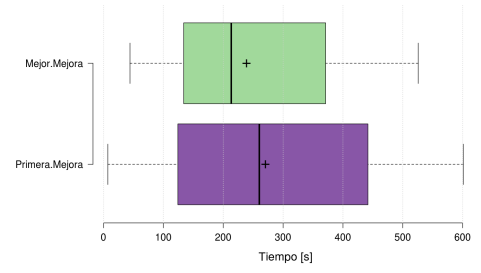
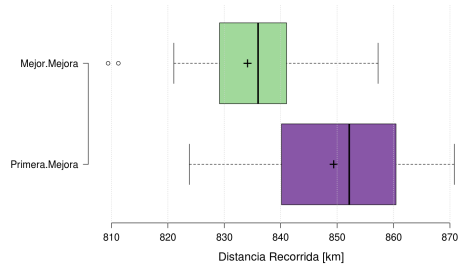
Para las instancias C9 y C10 el mínimo de distancia fue encontrado por primera mejora, pero las distribuciones fueron parejas para las dos categorías. Para C11, las distribuciones fueron similares pero mejor mejora obtuvo los mejores resultados. En el caso de C12, por otro lado, primera mejora muestra resultados considerablemente mejores. C9, C10 y C11 tienen la misma cantidad de puntos de demanda, pero aumentando la cantidad de depósitos (3, 4 y 5 respectivamente).

Caso

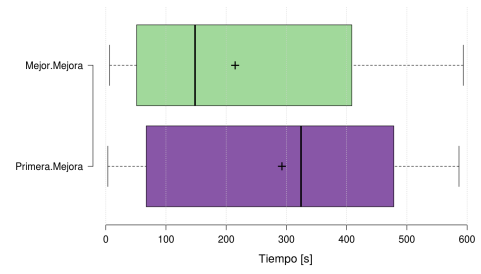
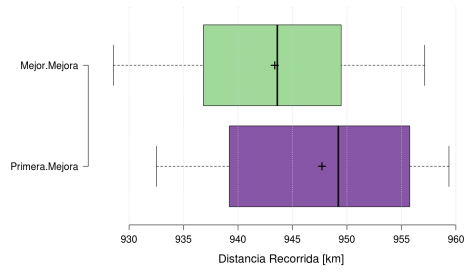
Distancia [km]

Tiempo [s]

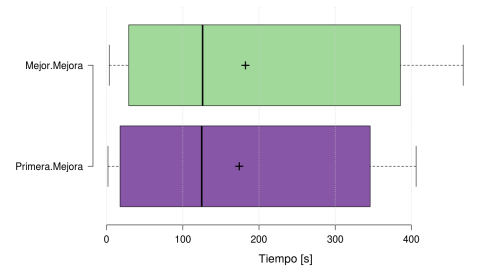
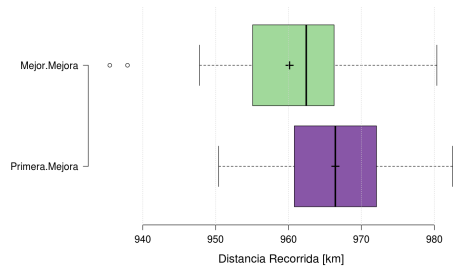
C5



C6



C7



C8

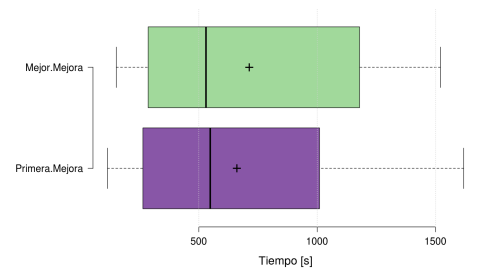
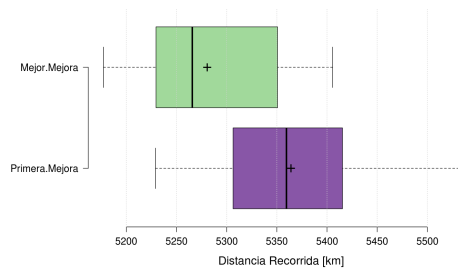


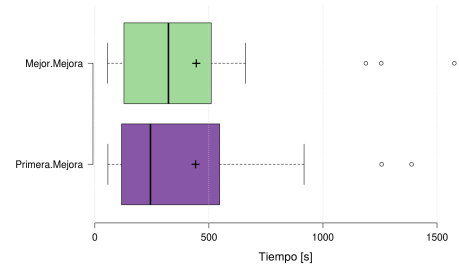
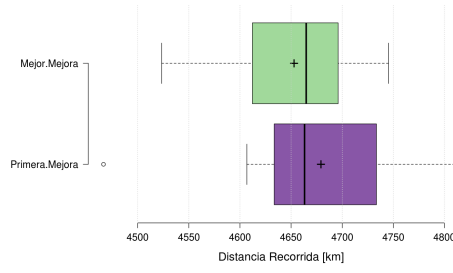
Figura 4.2: Comparación Búsqueda Local: Primera mejora versus Mejor Mejora – Instancias de Christofides: C5-C8

Caso

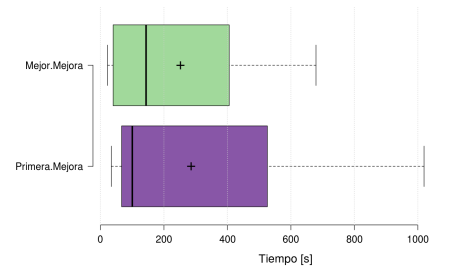
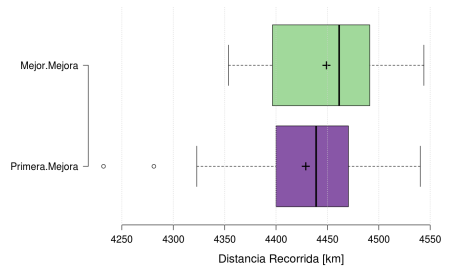
Distancia [km]

Tiempo [s]

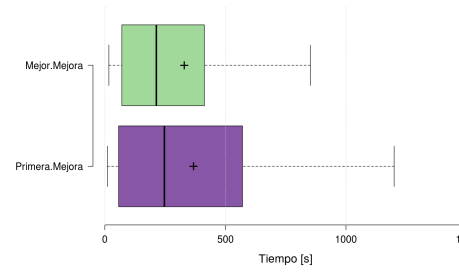
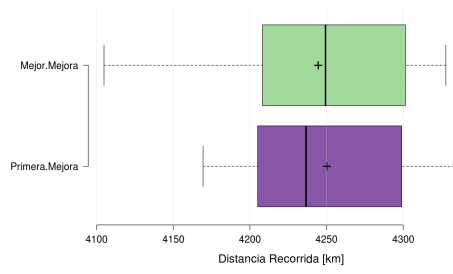
C9



C10



C11



C12

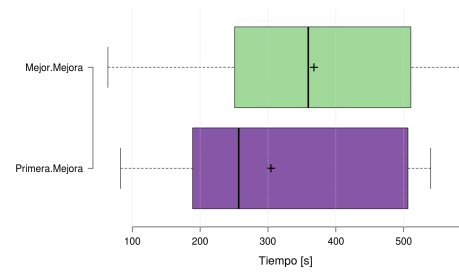
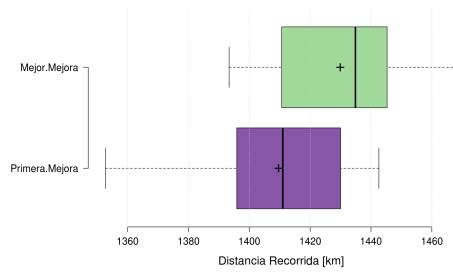


Figura 4.3: Comparación Búsqueda Local: Primera mejora versus Mejor Mejora – Instancias de Christofides: C9-C12

Debido a que el algoritmo propuesto incluye dividir el problema en varios problemas de enrutamiento, el hecho de asignar los puntos a un mayor número de depósitos hace que el tamaño del problema disminuya, otorgando a mejor mejora los mejores puntajes en C11. C9 y C10 tienen un número más alto de puntos por depósito, haciendo que el procesamiento sea más lento para mejor mejora y por lo tanto, utilizando detención por tiempo, dando la opción a primera mejora de realizar más iteraciones y explorar de mejor manera el universo de soluciones.

Finalmente, la figura 4.4 muestra los resultados comparativos de las instancias C15 y C18. Para la distancia, primera mejora fue capaz de obtener los mejores resultados en ambas instancias. Ambas instancias tienen un gran número de puntos de demanda, haciendo primera mejora una alternativa más eficiente de exploración del universo, contrastada con mejor mejora, que usa mucho tiempo nuevamente en ejecutar todas las variaciones de CROSS-Exchange, Or-Opt y Relocate. La distribución de tiempos de ejecución es similar, con mejor mejora teniendo una media más baja en ambos casos.

En esta sección se analizaron dos estrategias de búsqueda local: Primera Mejora, consistente en usar los operadores CROSS-Exchange, Relocate y Or-Opt pero detenerse si se encuentra una solución mejor que la original; o Mejor Mejora, consistente en usar los tres operadores para generar el vecindario completo de soluciones, y tomar la mejor.

Los resultados son mixtos, con algunas instancias (C3, C4, C5, C6, C7, C8, C9 y C11) mostrando resultados favorables para Mejor Mejora en términos de distancia, mientras que las otras (C1, C2, C10, C12, C15 y C18) presentan mejores resultados para Primera Mejora. Esto es esperable debido a los tamaños de los problemas. A medida que crece el número de puntos de demanda de las instancias, la supremacía entre mejor mejora y primera mejora varía. Inicialmente, para las instancias más pequeñas, mejor mejora tiene mejores resultados debido a que es capaz de cubrir el dominio completo en cada iteración, en lugar de conformarse con el primer resultado que presente alguna mejoría. Una vez que la cantidad de puntos de demanda aumenta las circunstancias cambian, y mejor mejora termina usando mucho tiempo de cómputo para compilar el vecindario completo, dando a primera mejora la ventaja de obtener un resultado mejor y avanzar en las iteraciones.

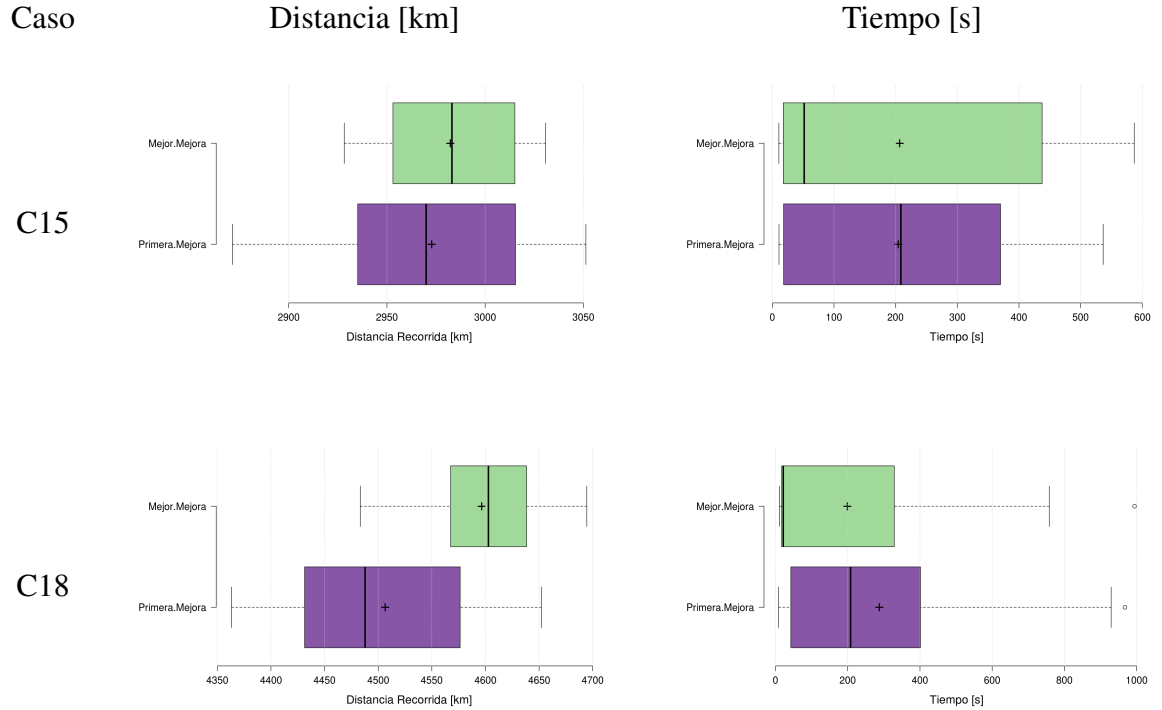


Figura 4.4: Comparación Búsqueda Local: Primera mejora versus Mejor Mejora – Instancias de Christofides: C15-C18

4.3.2. Esquema de control de feromona

En esta sección se presentan los resultados comparativos de los tres esquemas de control de feromonas estudiados: Esquema MaxMin (propuesto en [44]), consistente en utilizar valores límite superiores e inferiores para las feromonas y actualizarlo en tiempo de ejecución de acuerdo a los mejores resultados encontrados por el algoritmo; MaxMin fijo, un concepto similar pero que no contempla actualizar los valores de $\tau_{ij}^{\min}$ y $\tau_{ij}^{\max}$, y configura los valores utilizando los mejores resultados encontrados en la literatura para cada una de las instancias; y una variación de MaxMin fijo, representada como MaxMin Unificado, que consta en usar límites fijos para $\tau_{ij}^{\min}$ y $\tau_{ij}^{\max}$, pero equivalentes para todas las instancias. Con el fin de comparar, para MaxMin variable se utilizaron los resultados de mejor mejora de la sección de búsqueda local, nuevamente representados por color verde claro. MaxMin fijo es representado por el color naranja y MaxMin unificado es representado por el color azul.

En la figura 4.5 se observan los resultados de las instancias C1 a C3. En las tres instancias se puede observar que MaxMin Fijo, usando límites fijos distintos según la instancia que se está ejecutando, fue la que arrojó mejores resultados de distancia, sin embargo las diferencias entre los resultados para C1 son menores (las tres estrategias entregaron resultados dentro de un rango muy pequeño). Las medias de las muestras fueron parecidas entre las estrategias para las 3 instancias, salvo en C3 donde MaxMin unificado tuvo resultados claramente inferiores.

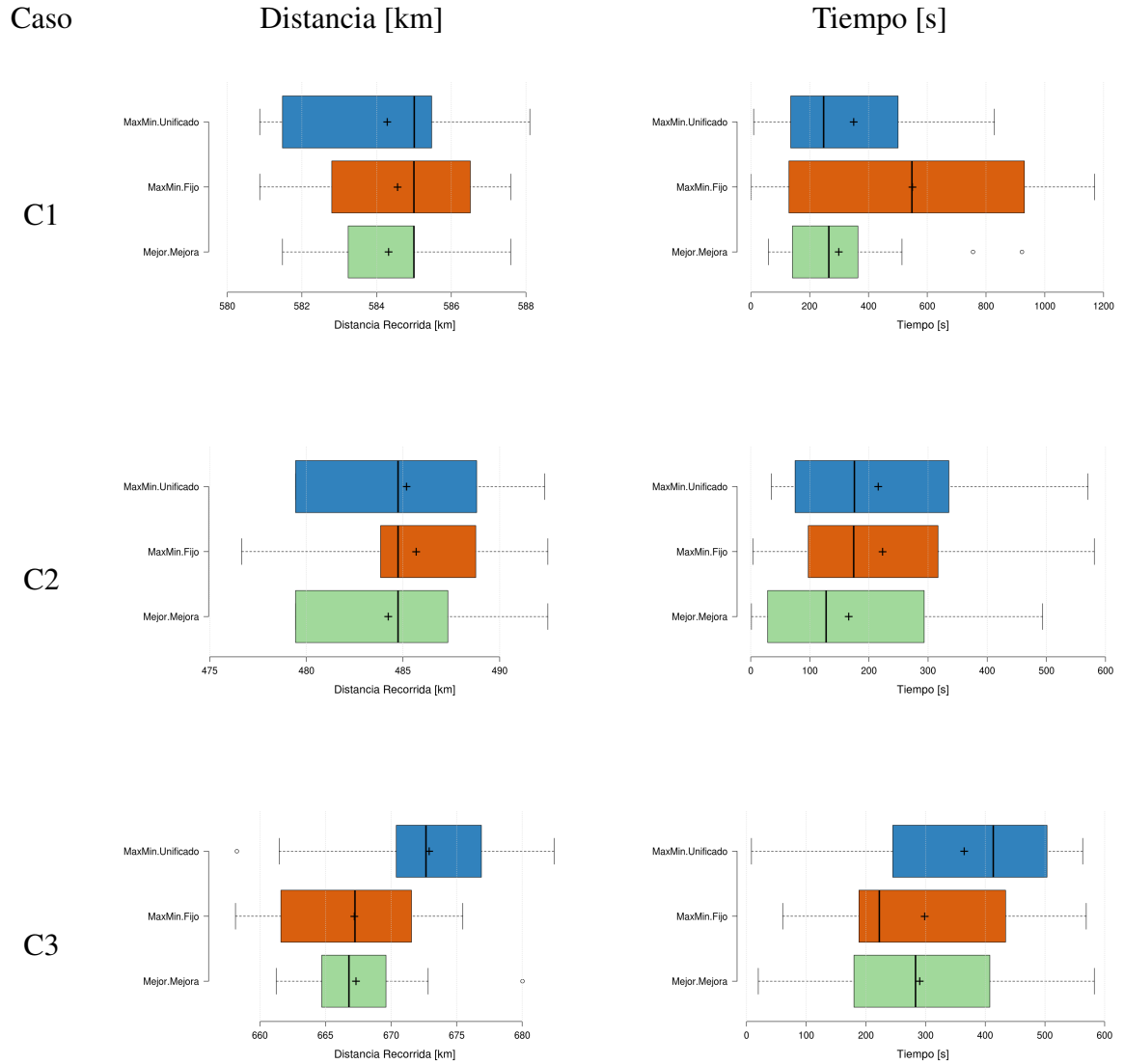


Figura 4.5: Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C1-C3

Respecto a los tiempos de ejecución, Se observa que para las 3 instancias los rangos fueron parecidos, siendo C1 la instancia donde se observa una diferencia mayor con MaxMin fijo, con mayores tiempos de ejecución en general. Estos resultados son esperables ya que las instancias C1, C2 y C3 pueden ser consideradas instancias pequeñas (50 a 75 puntos de demanda). Si bien algunas estrategias tienen mejor rendimiento que otras en cada una de las instancias, los resultados son similares sin otorgar una supremacía clara a alguno de los métodos.

En la figura 4.6 se observan los resultados comparativos de las instancias C4 a C6. En C4 se observa que el resultado en distancia obtenido por MaxMin variable, en verde, tiene considerablemente mejor resultado que MaxMin fijo por instancia y MaxMin fijo unificado. Al avanzar en las instancias las estrategias se van igualando, con MaxMin fijo perdiendo su supremacía y termina en C6 con resultados parecidos entre las 3 estrategias, con MaxMin unificado obteniendo mínimos levemente mayores que MaxMin fijo, pero con menor varianza.

En cuanto a los tiempos, MaxMin unificado y MaxMin fijo presentan resultados similares en C4 y C5, menores que MaxMin regular, y finalmente en C6 los resultados de las tres estrategias son semejantes.

En la figura 4.7 se observan los resultados para las instancias C7 a C9. En términos del tiempo, las instancias se comportan de manera similar en C7. En C8 MaxMin variable tiene más varianza que el resto de las estrategias, pero lo opuesto ocurre para C9, donde MaxMin variable es el que tiene menor varianza de las 3.

En el caso de la distancia, en C7 los resultados están equilibrados y son muy similares entre las tres estrategias. Luego, en C8, los mínimos se observan en las estrategias de MaxMin fijas (azul y naranja) mientras que la media de la muestra es mejor para MaxMin variable. Finalmente en C9 los resultados vuelven a ser muy similares entre las tres estrategias.

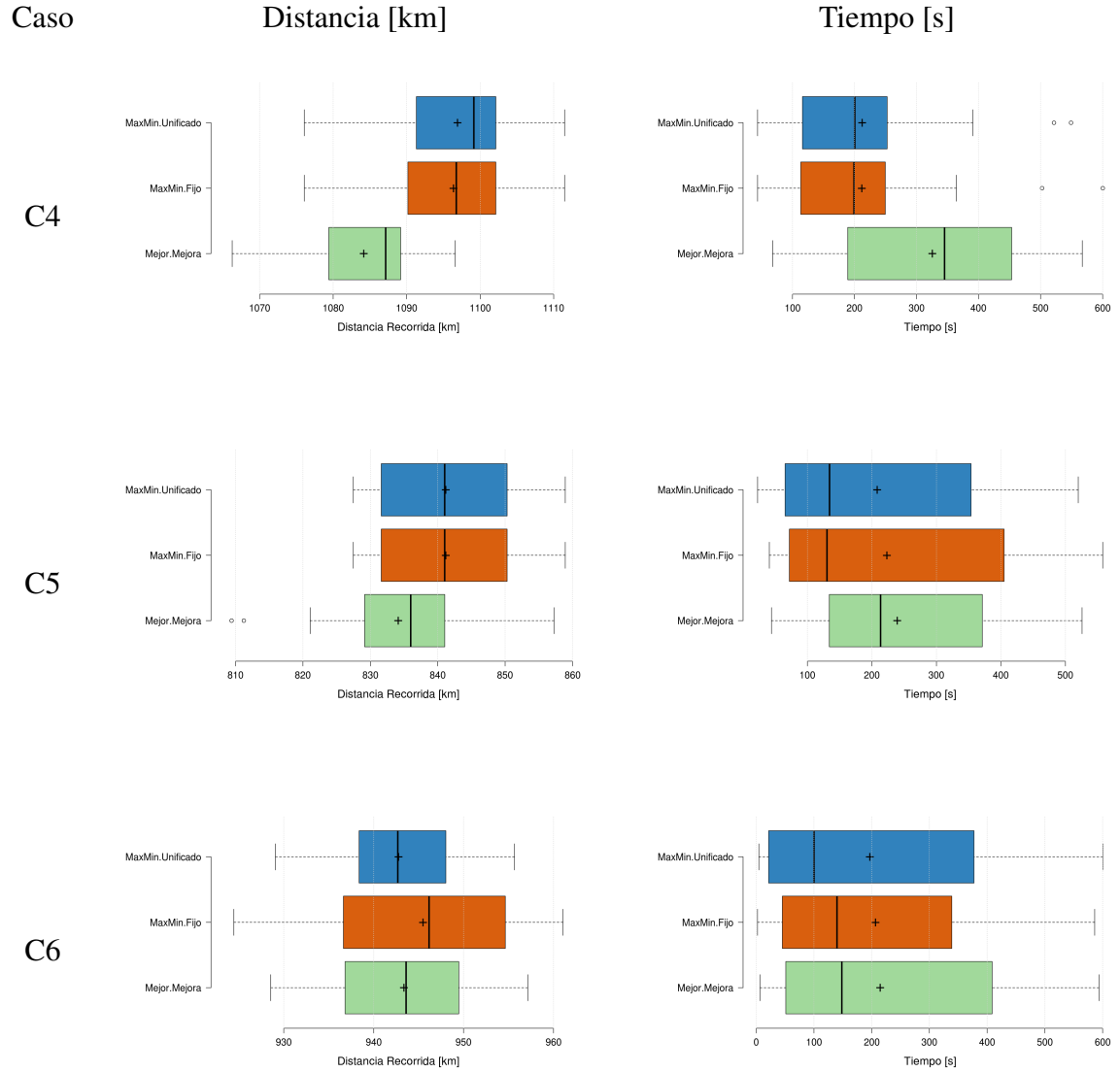


Figura 4.6: Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C4-C6

La figura 4.8 muestra los resultados comparativos de las instancias C10 a C12. En términos de tiempo, la estrategia de Max-Min variable solo tiene mejores resultados en C11, mientras que en C10 y C12 las versiones de Max-Min fijos tienen resultados más cortos.

En términos de distancia, por otro lado, Max-Min variable tiene consistentemente una distribución de resultados más favorable, pero los mínimos en C10 y C12 corresponden a las estrategias de Max-Min fijos, tanto dependiente de la instancia como unificado para todas las instancias.

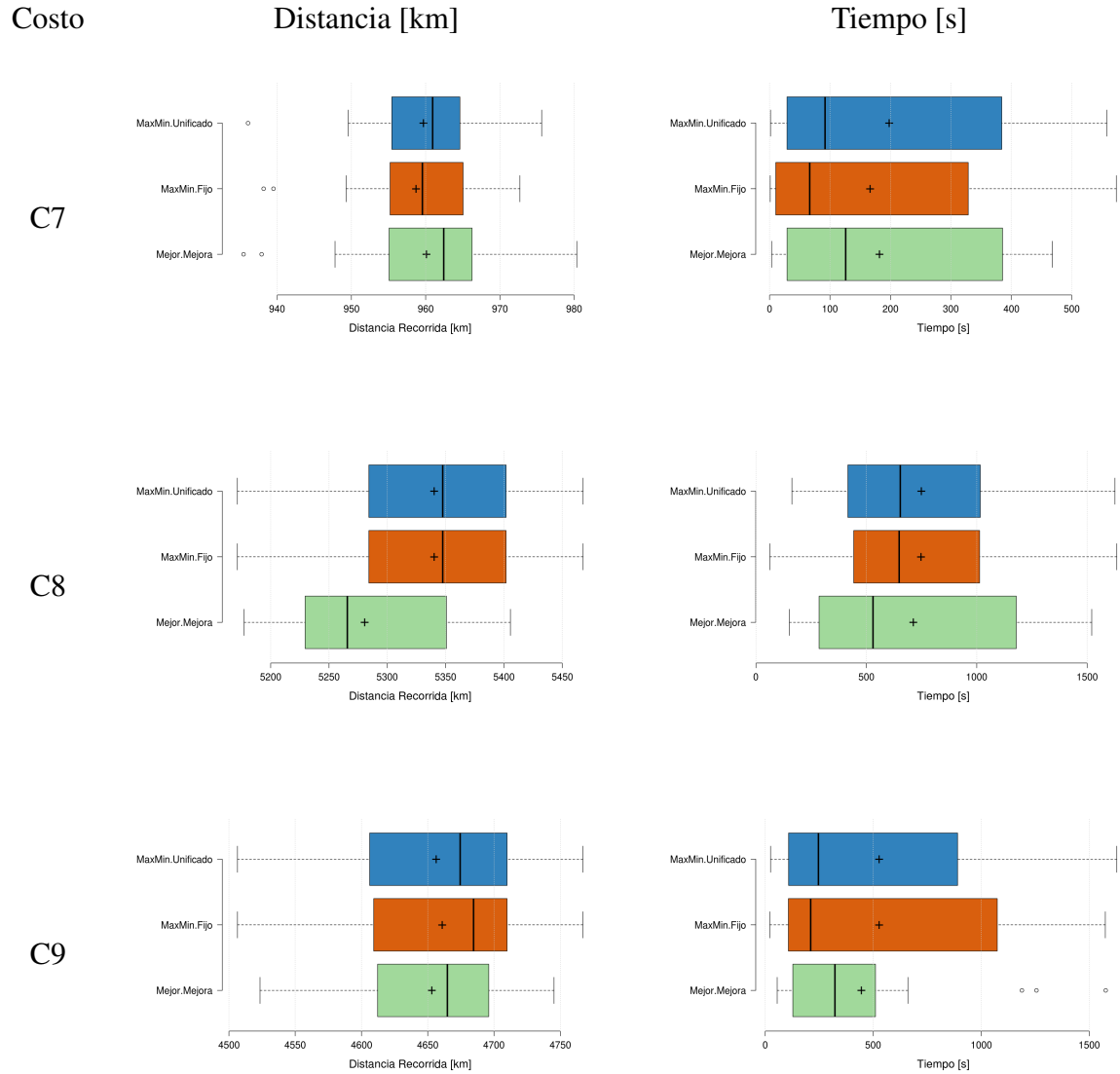


Figura 4.7: Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C7-C9

Finalmente, la figura 4.9 muestra los resultados comparativos para las instancias C15 y C18.

Para C15, los menores tiempos fueron obtenidos por Max-Min variable (equivalente a Mejor Mejora), pero la distribución muestra más varianza. Lo mismo ocurre para Max-Min fijo por instancia. Max-Min fijo unificado, con valores máximos y mínimos únicos para todas las instancias, presenta tiempos consistentemente más bajos. En C18 ocurre lo opuesto, y es Max-Min variable quien presenta menor varianza.

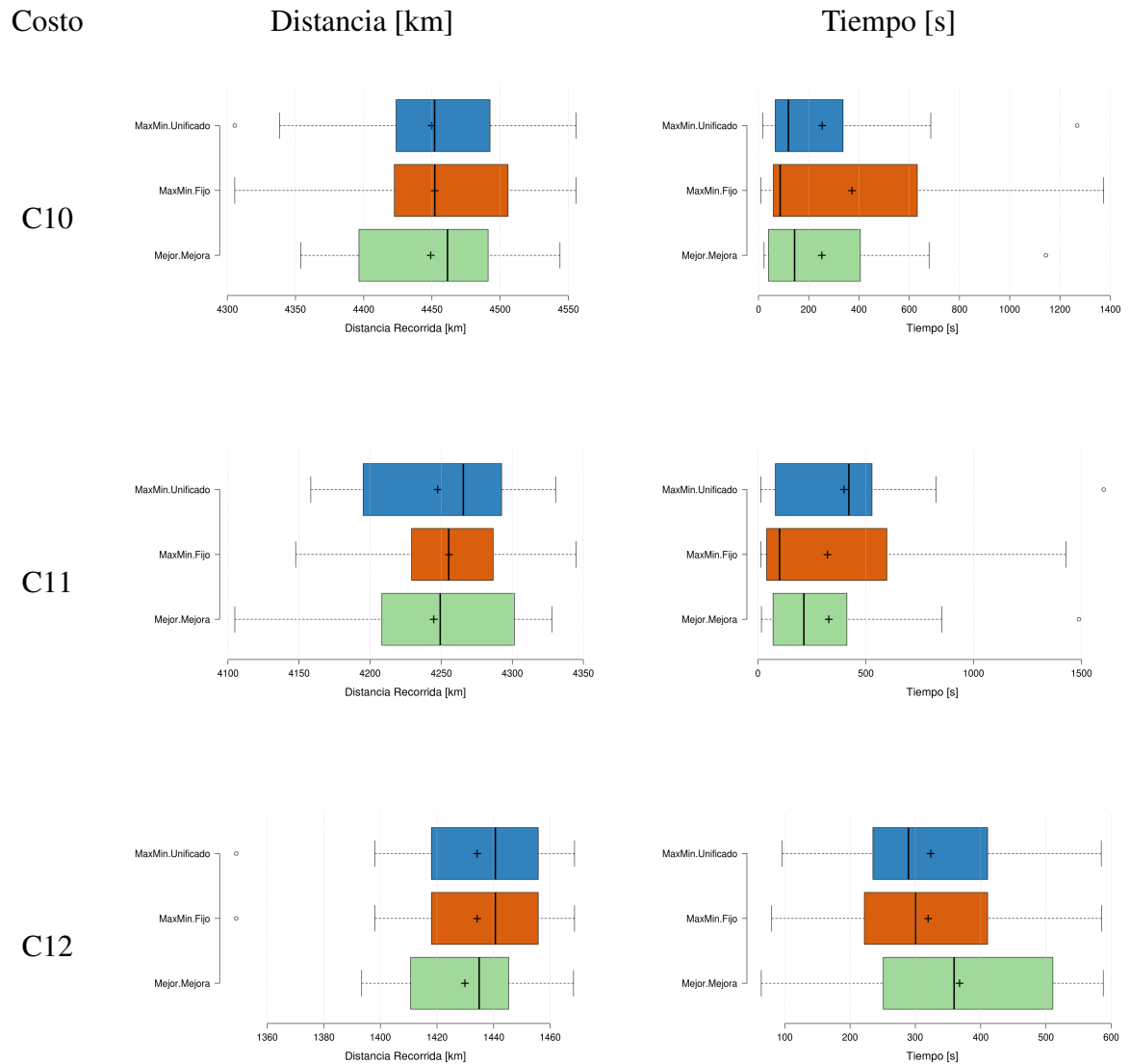


Figura 4.8: Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C10-C12

Para la distancia, en ambas instancias el grueso de las muestras fueron similares para los tres esquemas, con MaxMin fijo por instancia teniendo resultados mínimos mejores.

En esta sección se analizaron tres estrategias de control de feromonas: MaxMin variable, donde los valores de $\tau_{ij}^{\min}$ y $\tau_{ij}^{\max}$ cambian al avanzar el algoritmo; MaxMin fijo por instancia, donde se usan valores de $\tau_{ij}^{\min}$ y $\tau_{ij}^{\max}$ constantes dependientes de la instancia que se está ejecutando; y MaxMin fijo unificado, donde se usan valores de $\tau_{ij}^{\min}$ y $\tau_{ij}^{\max}$ fijos y constantes para todas las instancias. Los resultados nuevamente vuelven a ser mixtos, con 10 instancias (C1, C2, C3, C6, C8, C9, C10, C12, C15 y C18) para las cuales los esquemas MaxMin fijos obtuvieron mejores valores para distancia recorrida. Sin embargo, entre los dos algoritmos de MaxMin fijo no parece haber una supremacía clara. En términos de tiempo de ejecución, los resultados fueron similares entre MaxMin variable y MaxMin fijo unificado, siendo MaxMin fijo por instancia quien presentó los tiempos más largos.

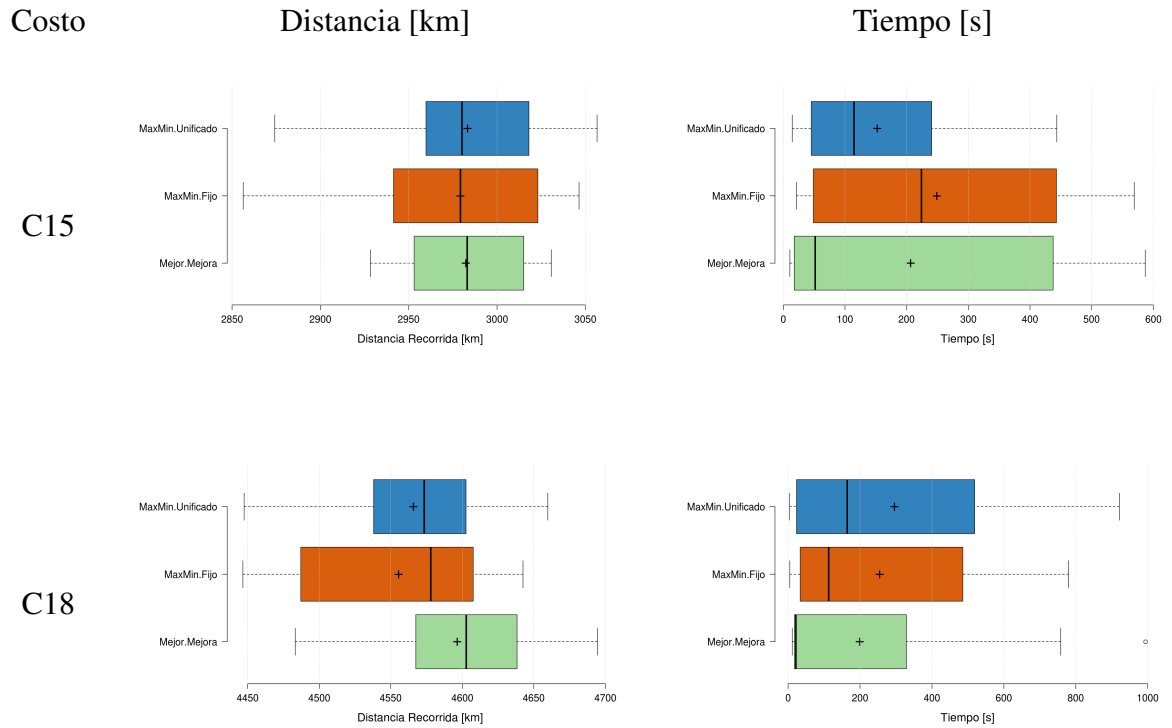


Figura 4.9: Comparación Control de Feromonas: MaxMin versus MaxMin fijo y variante – Instancias de Christofides C15-C18

4.3.3. Alternativa para operadores de búsqueda local

En esta sección se presentan los resultados comparativos de las alternativas para el uso de operadores en la búsqueda local: Uso de CROSS-Exchange, que se encarga de intercambiar tramos de las rutas entre sí, Or-Opt, encargado de desviar rutas a tramos originalmente pertenecientes a otras, y Relocate, que funciona trasladando puntos a lo largo de sus propias rutas a otras posiciones; y uso de un solo operador entre CROSS-Exchange, Or-Opt y Relocate, escogido al azar para cada iteración. En ambas versiones se explota el uso de los operadores, es decir, se aplica el operador para todos los elementos posibles de su dominio, usando todas las combinaciones de puntos y posiciones.

Las figuras 4.10 a 4.13 muestran para cada una de las instancias estudiadas, la calidad de las soluciones encontradas (en Km recorridos) y el tiempo de cómputo de los algoritmos comparados (en segundos). Para ambas versiones se usó una estrategia de Mejor Mejora, por lo que para la versión que ejecuta todos los operadores los resultados son equivalentes a los resultados de Mejor Mejora en la sección 4.3.1, nuevamente en verde claro. En rosa se presentan los resultados de el algoritmo usando un solo operador al azar en cada paso.

La figura 4.10 presenta los resultados para las instancias C1 a C4. Respecto al tiempo, para las instancias C1 a C3 los resultados indican que usar CROSS-Exchange, Or-Opt y Relocate obtuvo sus mejores resultados en tiempos menores que usar solo un operador al azar entre los tres. C4 por otro lado muestra tiempos menores para el uso de un operador al azar.

En términos de la distancia las diferencias son claras. Para las cuatro instancias los mejores resultados fueron los encontrados por el uso de todos los operadores, dando cuenta del efecto que el uso combinado tiene en los resultados de las instancias más pequeñas (C1 y C2 tienen 50 puntos de demanda, C3 tiene 75). En C4, con 100 puntos de demanda, la diferencia de distancia recorrida entre los dos algoritmos es substancial. Esto se debe a que, mientras más grande la instancia, el usar más operadores para agregar variedad y exploración al vecindario es más importante.

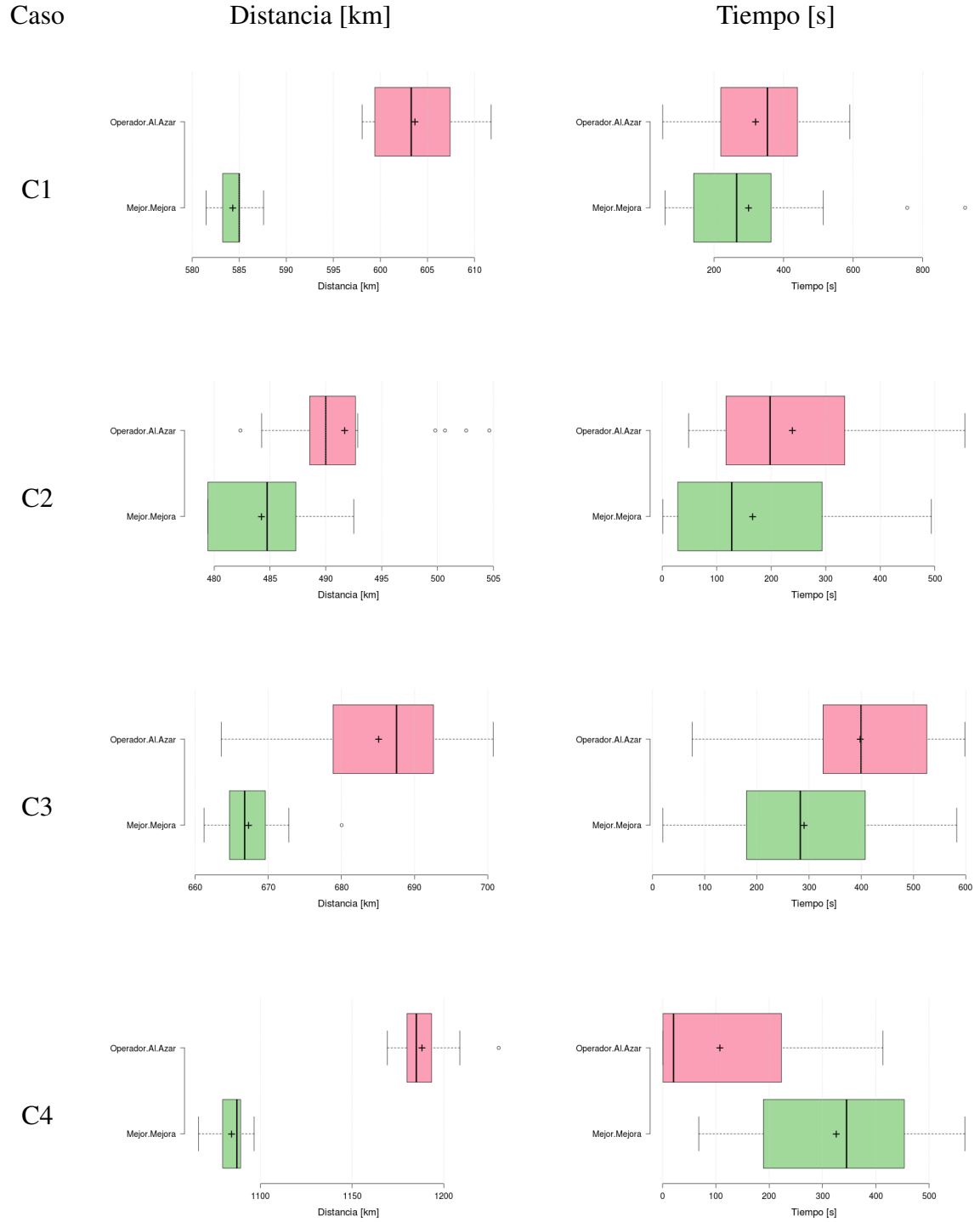


Figura 4.10: Comparación Operadores en Búsqueda local: Operador al azar versus todos los operadores (Mejor Mejora) – Instancias de Christofides: C1 - C4

En la figura 4.11 se observan los resultados para las instancias C5 a C8. En estas instancias la tendencia se repite: los tiempos de ejecución son menores para la versión del algoritmo con operador al azar, y en C7 incluso se observa que la distribución es muy pequeña. Esto es indicativo de que el algoritmo se estancó rápidamente, sin poder obtener mejores resultados luego de los primeros segundos de ejecución. En términos de la distancia, también se observan los mismos casos, donde operador al azar obtiene resultados considerablemente más altos que el uso de todos los operadores.

Esta tendencia se observa porque, entre C5 y C7, el tamaño de las instancias es mediano (100 puntos de demanda). Para C8, la instancia más grande del conjunto con 249 puntos de demanda, la diferencia es aún más significativa. A medida que el tamaño aumenta, la versión con operador al azar pierde competitividad con el algoritmo usando CROSS-Exchange, Or-opt y Relocate.

En la figura 4.12 presenta los resultados comparativos de las instancias C9 a C12. Para la distancia, la tendencia se mantiene: C9 y C10 muestran resultados significativamente peores para la versión con operador al azar, y esta diferencia mejora levemente para C11, y es aún menor para C12.

En términos de tiempo, en C9 se observa que la estrategia de operador al azar se estanca más rápidamente que el usar Or-Opt, CROSS-Exchange y Relocate, pero la diferencia no es significativa. Para C10 y C11 en cambio, el estancamiento es substancial, y la estrategia de operador aleatorio prácticamente no puede encontrar mejores soluciones luego de pasar un segundo de ejecución. En C12, al igual que para la distancia, la situación se normaliza y los tiempos para ambas estrategias son similares. Las diferencias drásticas que se observan en C9, C10 y C11 se explican debido a que son las instancias más grandes de la muestra, con 249 puntos de demanda. En C12, en cambio, el tamaño disminuye considerablemente (a 80 puntos de demanda), reflejado en los resultados menos extremos para esa instancia.

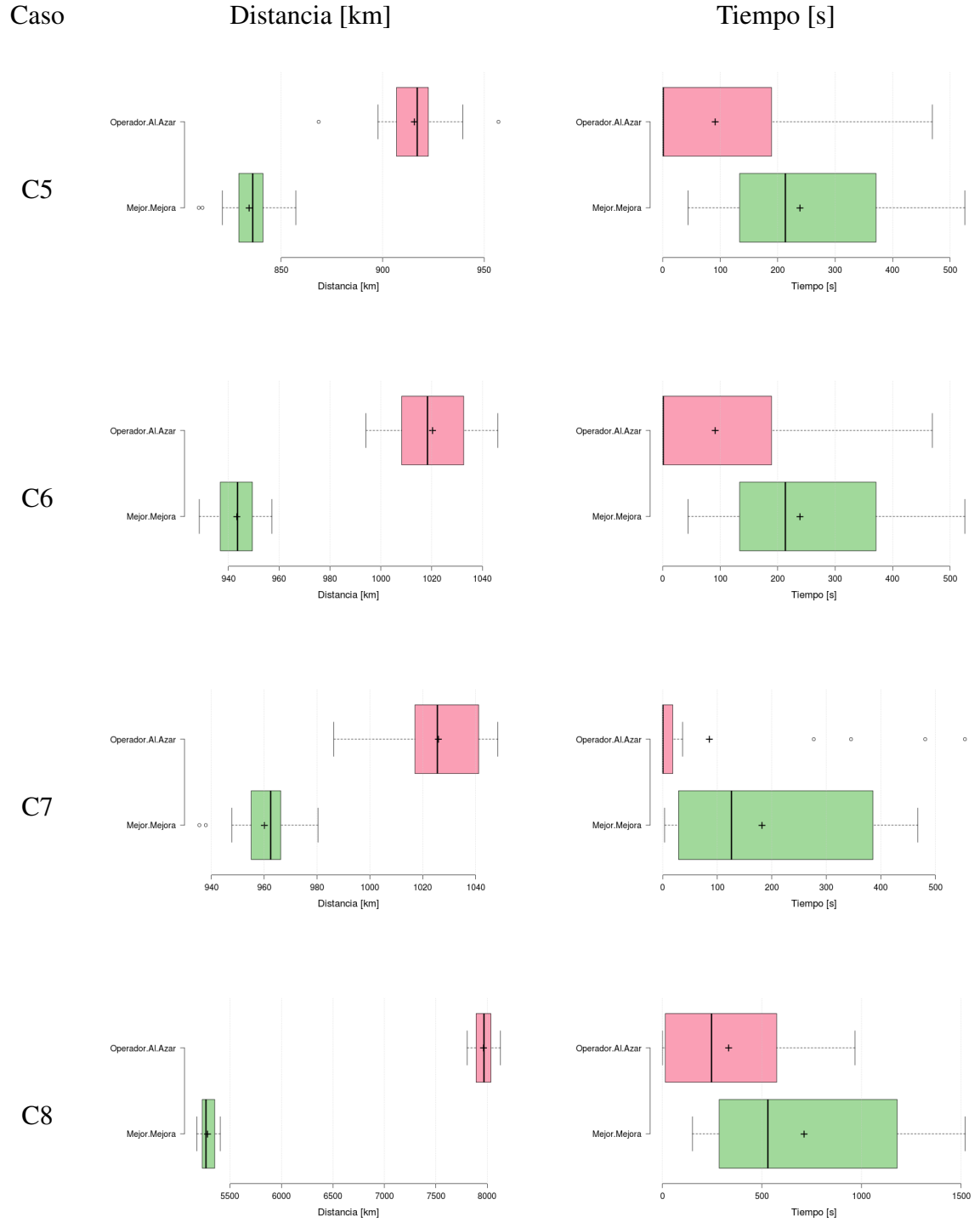


Figura 4.11: Comparación Operadores en Búsqueda local: Operador al azar versus todos los operadores (Mejor Mejora) – Instancias de Christofides: C5 - C8

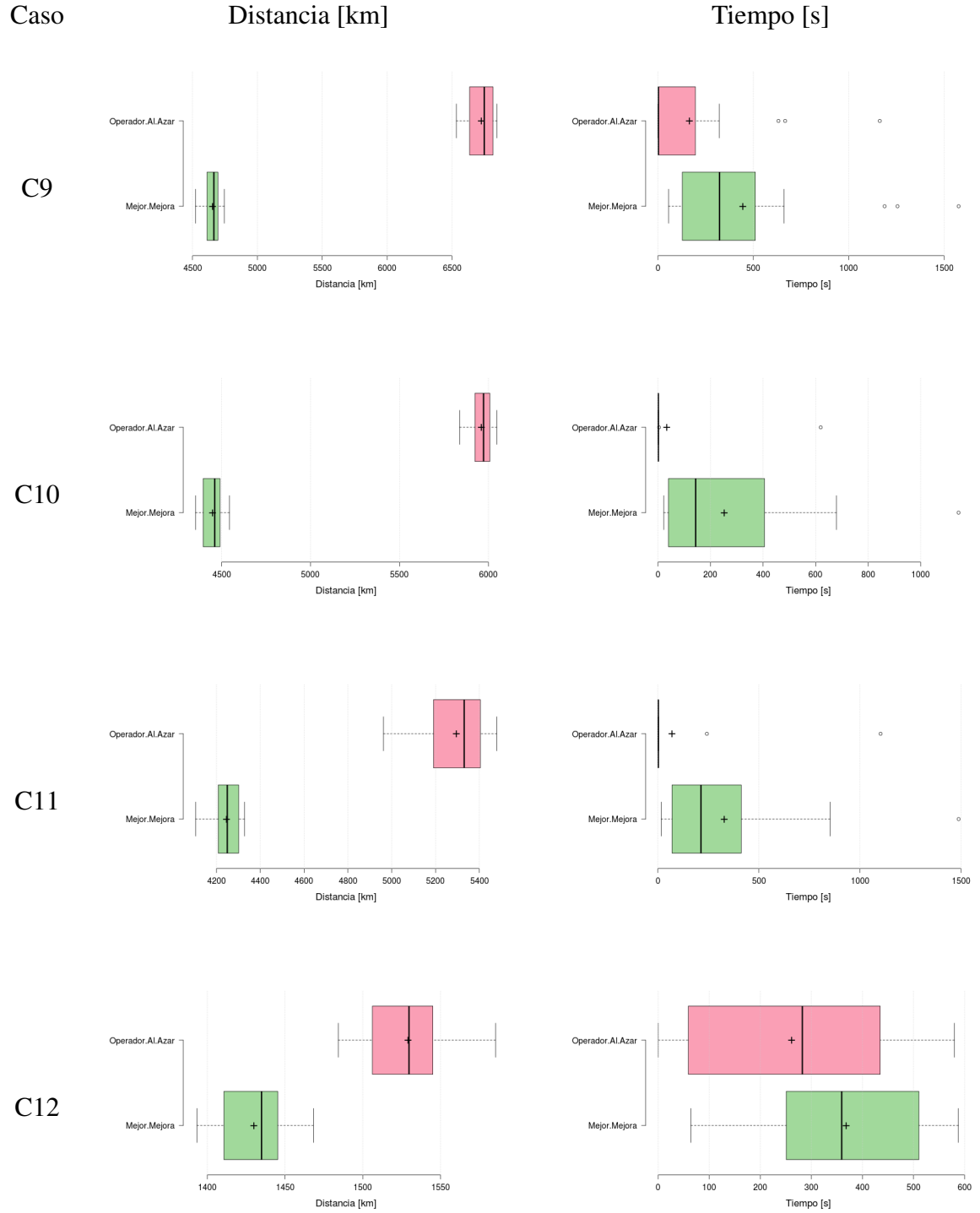


Figura 4.12: Comparación Operadores en Búsqueda local: Operador al azar versus todos los operadores (Mejor Mejora) – Instancias de Christofides: C9 - C12

Finalmente, en la figura 4.13 se observan los resultados para las instancias C15 y C18. En términos del tiempo, para ambas instancias se observan resultados similares, con medias parecidas. Para la distancia recorrida, el uso de los operadores CROSS-Exchange, Or-Opt y Relocate en conjunto entrega resultados mejores que usar solo uno de esos operadores al azar, pero la diferencia no es drástica como en otras instancias anteriores. Ambas instancias tienen un gran número de puntos de demanda (C15 tiene 160 y C18 tiene 240 puntos), pero el número de depósitos y por ende, el número de puntos de demanda por depósito, se mantiene bajo, con 40 puntos de demanda por depósito para ambos casos, lo que explica lo similar de los resultados.

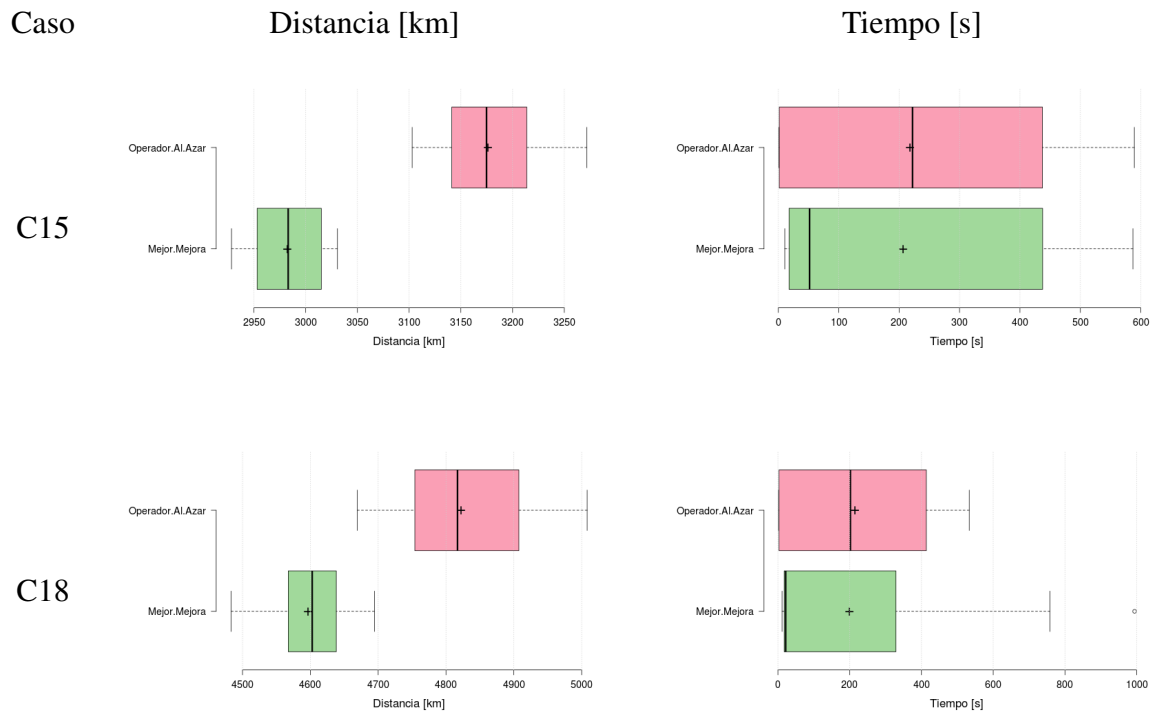


Figura 4.13: Comparación Operadores en Búsqueda local: Operador al azar versus todos los operadores (Mejor Mejora) – Instancias de Christofides: C15 - C18

En esta sección se analizaron dos alternativas de uso de operadores para la etapa de la búsqueda local: Usar CROSS-Exchange, Or-Opt y Relocate; y usar solo uno de los operadores entre

CROSS-Exchange, Or-Opt o Relocate, elegido al azar en cada iteración. Los resultados comparativos fueron drásticos: la estrategia de operador aleatorio obtuvo peores resultados para todas las instancias, y en el caso de las instancias más grandes, la diferencia fue extrema. En términos de los tiempos de ejecución, para las instancias más grandes los tiempos usados por la estrategia de operador aleatorio fueron significativamente menores, mostrando un estancamiento luego de unos segundos de ejecución.

4.3.4. Comparación con otros trabajos del estado del arte

Para comparar el algoritmo con otras alternativas presentadas en el estado del arte, se elige un conjunto de soluciones de referencia compuestas por las siguientes proposiciones: TabuSearch, propuesto por Cordeau, Gendreau y Laporte en [10] y representada por la sigla TS; el algoritmo genético propuesto por Ombuki-Berman et al. en [36], representado por GA; GenClust, propuesto por Thangiah et al. en [47]; y el algoritmo de optimización por colonia de hormigas propuesto por Stodola en [43], representado por ACO Stodola. Este conjunto de soluciones presentan una variedad de estrategias que permite comparar el desempeño del algoritmo propuesto, denotado por ACO Mejor Mejora, dando la oportunidad de observar el comportamiento del algoritmo versus distintas clases de soluciones. Los resultados entregados por cada uno de los algoritmos se presentan en kilómetros, debajo del nombre del algoritmo, y el tiempo usado en entregar dichas soluciones se presenta en segundos, bajo las columnas t [s] a la derecha de cada algoritmo. Las instancias son etiquetadas con su número y el número de puntos de demanda (n) versus el número de depósitos (d), y debajo de cada una se presenta también el error relativo frente al mejor resultado obtenido en la literatura. La comparación se puede apreciar en la tabla 4.4. Los tiempos marcados como N. P. no fueron publicados en la investigación original.

En la tabla 4.4 se observa que los resultados del algoritmo son competitivos para las primeras instancias (C1 a C7), las cuales tienen los tamaños más pequeños (50 puntos de demanda para C1 y C2, 75 puntos de demanda para C3, 100 puntos de demanda para C4-C7). Como se apreció en las secciones anteriores, cuando el tamaño de la instancia supera los 100 puntos de demanda, los resultados empeoran debido a que al cantidad de puntos hace que sea más

Instancia (n/d)	TS [km]	t [s]	GA [km]	t [s]	GenClust [km]	t [s]	ACO Stodola [km]	t [s]	ACO Mejor Mejora [km]	t [s]
C1 (50/4)	576.9	194.4	622.9	N. P.	591.7	336	576.9	1.5	581.5	60.3
Error	0.00 %		7.9 %		2.6 %		0.00 %		0.8 %	
C2 (50/4)	473.9	207.6	480.04	N. P.	483.1	186	475.9	1.6	479.4	288.6
Error	0.0 %		1.3 %		1.9 %		0.4 %		1.2 %	
C3 (75/5)	645.5	339.6	706.9	N. P.	694.5	264	645.5	3.7	661.2	420.0
Error	0.00 %		9.5 %		7.6 %		0.00 %		2.4 %	
C4 (100/2)	1006.5	467.4	1024.8	N. P.	1062.4	504	1018.5	23.1	1066.3	558.3
Error	0.00 %		1.8 %		5.6 %		1.2 %		5.9 %	
C5 (100/2)	753.3	492.6	785.2	N. P.	755.8	618	755.7	20.7	809.4	386.4
Error	0.00 %		4.2 %		0.3 %		0.3 %		7.5 %	
C6 (100/3)	877.8	459	908.9	N. P.	976.0	582	885.4	25.7	928.6	83.0
Error	0.00 %		3.5 %		11.2 %		0.9 %		5.8 %	
C7 (100/4)	892	462.2	918.1	N. P.	976.5	444	895.5	25.2	935.5	447.8
Error	0.0 %		2.9 %		9.5 %		0.4 %		4.9 %	
C8 (249/2)	4482.4	1525.8	4690.2	N. P.	4,812.5	2508	4482.4	233.1	5177.2	186.8
Error	0.00 %		4.6 %		7.4 %		0.0 %		15.5 %	
C9 (249/3)	3920.9	419.4	4,240.1	N. P.	4,284.6	1884	3990.2	1008	4523.5	403.1
Error	0.0 %		8.1 %		9.3 %		1.8 %		15.4 %	
C10 (249/4)	3714.7	984	3984.8	N. P.	4,291.5	2730	3751.5	876	4353.8	117.9
Error	0.0 %		7.3 %		15.5 %		0.9 %		17.2 %	
C11 (249/5)	3580.4	1040	3880.7	N. P.	4,092.7	2244	3657.2	948	4104.8	445.9
Error	0.0 %		8.4 %		14.3 %		2.1 %		14.7 %	
C12 (80/2)	1318.9	334	1318.9	N. P.	1,421.9	288	1318.9	8.9	1393.4	349.0
Error	0.0 %		0.0 %		7.8 %		0.00 %		5.6 %	
C15 (160/4)	2505.4	844	2579.3	N. P.	3,059.2	474	2510.1	78	2928.3	74.1
Error	0.0 %		2.6 %		22.1 %		0.2 %		16.9 %	
C18 (240/6)	3702.9	1455	3903.9	N. P.	5,462.9	1218	3741.8	690	4483.3	262.3
Error	0.0 %		5.4 %		47.5 %		1.1 %		21.1 %	

Cuadro 4.4: Comparación de trabajos del estado del arte con el algoritmo propuesto

extenso en tiempo explorar áreas mayores de los vecindarios de soluciones. Es el caso de C8 a C11, donde en todas las instancias el error encontrado respecto al mejor resultado de la literatura fue mayor a 14.6 %. Para las instancias C12 a C18, en las cuales el tamaño de la instancia varía entre 80 y 240 puntos de demanda, se aprecia que los resultados vuelven a ser mejores, incluso superando a algunos de los algoritmos de la literatura, debido a que la cantidad de depósitos amortizan el tamaño completo de la instancia.

Respecto al desempeño del algoritmo frente a sus pares de la literatura, se observa que para la mayor parte de los casos el algoritmo propuesto se comporta de similar manera a GenClust, propuesto en [47], presentando mejores resultados que este algoritmo en las instancias C1, C2, C3, C6, C7, C12, C15 y C18. La dependencia de la exploración en la etapa de búsqueda local del algoritmo propuesto perjudica el rendimiento a la hora de compararlo con el algoritmo por colonización de hormigas propuesto por Stodola en [43], alternativa que supera al algoritmo propuesto en todas las instancias. El algoritmo genético, GA, presentado en [45], tiene resultados mixtos comparados con el algoritmo propuesto: en instancias más pequeñas (C1-C3), ACO Mejor Mejora tiene un mejor rendimiento, pero esta realidad cambia a partir de la instancia C4, donde los mejores resultados son obtenidos por GA.

En términos de tiempo, debido a la cantidad de iteraciones ejecutadas y la dependencia de la etapa de búsqueda local, el algoritmo propuesto tiene un desempeño más lento que el algoritmo por colonia de hormigas propuesto por Stodola en [43] y que el algoritmo genético propuesto por Ombuki-Berman et al. en [36], pero tiene un mejor tiempo de ejecución que GenClust propuesto en [47].

Capítulo 5

Conclusiones

Luego de una catástrofe natural, el tiempo de respuesta a la hora de entregar suministros a las personas afectadas es vital para evitar enfermedades y problemas más severos en los albergues y comunidades aisladas. Muchas veces los equipos de emergencia disponen de recursos almacenados en múltiples centros de acopio, vehículos e información sobre cantidades aproximadas de personas estacionadas en cada uno de los albergues, por lo que la distribución de dichos recursos de modo de optimizar el tiempo de entrega y el buen uso de los materiales se puede modelar como un problema de enrutamiento de vehículos con múltiples depósitos.

En este trabajo se propone un algoritmo de optimización por colonias de hormigas para resolver este problema y entregar rutas que cumplan con las restricciones de la situación planteada. Este algoritmo fue implementado en un programa computacional para ser probado en un marco de referencia estándar. Para esto se usó un conjunto de 15 instancias propuestas por Christofides en [8] de entre 50 y 249 puntos de demanda. Esta implementación fue luego sintonizada a través del método paramILS. En este trabajo se presenta un análisis experimental de la propuesta analizando los resultados del algoritmo utilizando variaciones de estrategias de búsqueda local y de control de feromonas. Las variedades comparadas fueron las siguientes: estrategia de mejor mejora versus primera mejora; control de feromonas MaxMin variable versus dos variedades de MaxMin fijo; y el uso de un solo operador de búsqueda local al azar versus el uso de todos los operadores.

El primer análisis consistió en utilizar la estrategia de mejor mejora, la cual utiliza los operadores CROSS-Exchange, Relocate y Or-Opt para generar el vecindario completo de soluciones, comparada a la estrategia de primera mejora, la cual aplica los mismos operadores pero deteniendo la búsqueda con la primera solución vecina que tuviera un costo menor que la original. Los resultados del análisis indican que mejor mejora tiene un rendimiento superior respecto a primera mejora para la mayor parte de las soluciones, y que esta supremacía cambia a medida que varía el tamaño de las instancias. Para las instancias más pequeñas, de hasta 100 puntos de demanda, los resultados fueron favorables para mejor mejora, mientras que en las instancias con mayor número de puntos de demanda primera mejora domina. El tamaño de la instancia dicta que tantas soluciones vecinas se pueden generar a partir de los operadores de búsqueda local, por lo que mejor mejora, que realiza una búsqueda extensa del vecindario, tomará más tiempo en calcular todos los vecinos para instancias más grandes. Esto entrega la ventaja a primera mejora, que si bien no se va a detener necesariamente en el movimiento óptimo, podrá avanzar más rápido y seguir explorando el conjunto de soluciones posibles.

El segundo análisis consistió en utilizar el esquema de control de feromonas MaxMin variable, para el cual el algoritmo varía los valores máximos y mínimos que pueden tener las feromonas a medida que se encuentran mejores soluciones en cada iteración, y compararlo con dos variedades de MaxMin fijos, la primera usando un valor máximo y mínimo determinado por el mejor resultado encontrado en la literatura (dependiente de cada iteración) y la segunda utilizando valores máximos y mínimos fijos e iguales para todas las instancias. Los resultados indican que MaxMin variable fue la estrategia con mejor consistencia del análisis, y que la capacidad de adaptarse y variar la ventana de feromonas otorgan una ventaja frente a los otros dos esquemas de control de feromonas, que se mantienen inmóviles con respecto al progreso del algoritmo.

Finalmente se evaluaron dos alternativas de búsqueda local, una de las cuales ejecuta los operadores CROSS-Exchange, Relocate y Or-Opt y la otra que, en vez de utilizarlos todos, decide previamente a la etapa de búsqueda cual utilizar de manera aleatoria. Los resultados del análisis muestran que el uso de los tres operadores en conjunto es superior al uso de solo uno al azar. Esta diferencia es significativamente mayor en instancias de mayor tamaño

(75 puntos de demanda en adelante), ya que el uso de un solo operador al azar no permite observar de manera clara el vecindario de soluciones, factor crítico cuando se está tratando con tantas combinaciones y posibilidades en estas instancias grandes.

De este modo, se concluye que la mejor versión de este algoritmo contempla el uso de los operadores CROSS-Exchange, Or-opt y Relocate en conjunto en todas las iteraciones y utilizando una estrategia de mejor mejora. Para el esquema de feromonas, la mejor versión del algoritmo es aquella que usa MaxMin variable para determinar los límites mayores y menores de los valores de feromona.

Se compararon también los resultados obtenidos con otros algoritmos de la literatura, particularmente GenClust propuesto en [47], un algoritmo genético propuesto en [36], TabuSearch propuesto en [22] y un algoritmo por optimización de colonias de hormigas propuesto en [43]. Los resultados fueron competitivos, con el algoritmo propuesto teniendo mejores resultados que algunos de los otros en las instancias de menor tamaño (50 a 100 puntos de demanda, o mayor número de puntos de demanda pero también mayor número de depósitos), pero perdiendo desempeño en las instancias más grandes (249 puntos de demanda). Esto es causado por la dependencia en búsqueda local, ya que debido a que el algoritmo necesita los tres operadores para funcionar correctamente, el tiempo que tarda en ejecutar la etapa de búsqueda local crece exponencialmente al aumentar los puntos de demanda por depósito. Si bien las diferencias con los mejores resultados de la literatura son de tamaño considerable, esto indica que existe trabajo que realizar a futuro y el resultado de este trabajo fue un acercamiento positivo a encontrar una alternativa de algoritmos para las circunstancias de desastres naturales.

Se concluye finalmente que el algoritmo propuesto es un aporte positivo para resolver el problema de enrutamiento de vehículos considerando múltiples depósitos, y que cumple con los requisitos para ser una herramienta útil a la hora de responder de manera rápida y eficiente en una situación de emergencia.

Bibliografía

- [1] Michel L Balinski and Richard E Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.
- [2] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.
- [3] Bernd Bullnheimer, Richard F Hartl, and Christine Strauss. Applying the ant system to the vehicle routing problem. In *Meta-heuristics*, pages 285–296. Springer, 1999.
- [4] I-Ming Chao, Bruce L Golden, and Edward Wasil. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences*, 13(3-4):371–406, 1993.
- [5] Boris V Cherkassky and Andrew V Goldberg. On implementing the push—relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.
- [6] Nicos Christofides. Combinatorial optimization. In *A Wiley-Interscience Publication, Based on a series of lectures, given at the Summer School in Combinatorial Optimization, held in Sogesta, Italy, May 30th-June 11th, 1977, Chichester: Wiley, 1979, edited by Christofides, Nicos*, 1979.
- [7] Nicos Christofides and Samuel Eilon. An algorithm for the vehicle-dispatching problem. *Or*, pages 309–318, 1969.
- [8] Nicos Christofides, Aristide Mingozzi, and Paolo Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282, 1981.
- [9] Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.
- [10] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

- [11] Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 176–190. Springer, 2000.
- [12] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.
- [13] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.
- [14] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: An autocatalytic optimizing process. 1991.
- [15] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.
- [16] Samuel Eilon, CDT Watson-Gandy, Nicos Christofides, and Richard de Neufville. Distribution management-mathematical modelling and practical analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 4(6):589–589, 1974.
- [17] Marshall L Fisher and Ramchandran Jaikumar. *A decomposition algorithm for large-scale vehicle routing*. Department of Decision Sciences, Wharton School, University of Pennsylvania, 1978.
- [18] United Nations Office for Disaster Risk Reduction. Disaster statistics, 2014.
- [19] Centre for Research on the Epidemiology of Disasters CRED. Country profile, 2016.
- [20] Brian A Foster and David M Ryan. An integer programming approach to the vehicle scheduling problem. *Journal of the Operational Research Society*, 27(2):367–384, 1976.
- [21] Michel Gendreau, Alain Hertz, and Gilbert Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992.
- [22] Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290, 1994.
- [23] Billy E Gillett and Jerry G Johnson. Multi-terminal vehicle-dispatch algorithm. *Omega*, 4(6):711–718, 1976.
- [24] Billy E Gillett and Leland R Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349, 1974.

- [25] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.
- [26] Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.
- [27] Josiah Jacobsen-Grocott, Yi Mei, Gang Chen, and Mengjie Zhang. Evolving heuristics for dynamic vehicle routing with time windows using genetic programming. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 1948–1955. IEEE, 2017.
- [28] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59(3):345–358, 1992.
- [29] Gilbert Laporte, Hélène Mercure, and Yves Nobert. An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16(1):33–46, 1986.
- [30] Mirosław Malek, Mohan Guruswamy, Mihir Pandya, and Howard Owens. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research*, 21(1):59–84, 1989.
- [31] Yannis Marinakis and Magdalene Marinaki. A honey bees mating optimization algorithm for the open vehicle routing problem. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 101–108. ACM, 2011.
- [32] Yannis Marinakis, Athanasios Migdalas, and Panos M Pardalos. Expanding neighborhood grasp for the traveling salesman problem. *Computational Optimization and Applications*, 32(3):231–257, 2005.
- [33] Abdul Kadar Muhammad Masum, Mohammad Shahjalal, F Faruque, and IH Sarker. Solving the vehicle routing problem using genetic algorithm. *International Journal of Advanced Computer Science and Applications*, 2(7):126–131, 2011.
- [34] Roberto Montemanni, Luca Maria Gambardella, Andrea Emilio Rizzoli, and Alberto V Donati. Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343, 2005.
- [35] Volker Nannen and Agoston E Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *IJCAI*, volume 7, pages 975–980, 2007.
- [36] Beatrice Ombuki-Berman and Franklin T Hanshar. Using genetic algorithms for multi-depot vehicle routing. In *Bio-inspired algorithms for the vehicle routing problem*, pages 77–99. Springer, 2009.
- [37] Ilhan Or. Traveling salesman type combinatorial problems and their relation to the logistics of regional blood banking. 1977.

- [38] Christian Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.
- [39] Jin Qin, Yong Ye, Bi-rong Cheng, Xiaobo Zhao, and Linling Ni. The emergency vehicle routing problem with uncertain demand under sustainability environments. *Sustainability*, 9(2):288, 2017.
- [40] Dimitrios Sariklis and Susan Powell. A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, pages 564–573, 2000.
- [41] SERNAGEOMIN. Principales desastres ocurridos desde 1980 en chile. 2017.
- [42] Mojtaba Talebian Sharif and Majid Salari. A grasp algorithm for a humanitarian relief transportation problem. *Engineering Applications of Artificial Intelligence*, 41:259–269, 2015.
- [43] Petr Stodola and Jan Mazal. Applying the ant colony optimisation algorithm to the capacitated multi-depot vehicle routing problem. *International Journal of Bio-Inspired Computation*, 8(4):228–233, 2016.
- [44] Thomas Stützle and Holger H Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.
- [45] Paneerselvam Surekha and Sai Sumathi. Solution to multi-depot vehicle routing problem using genetic algorithms. *World Applied Programming*, 1(3):118–131, 2011.
- [46] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186, 1997.
- [47] Sam R Thangiah and Said Salhi. Genetic clustering: an adaptive heuristic for the multi-depot vehicle routing problem. *Applied Artificial Intelligence*, 15(4):361–383, 2001.
- [48] Wei Yi and Arun Kumar. Ant colony optimization for disaster relief operations. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):660–672, 2007.