UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

Una metaheurística para el problema de planificación de ruta más corta multipunto

POR

Esteban Ignacio Vallejos Yévenes

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Industrial

Profesor Guía
Carlos Contreras Bolton

Marzo 2022

Concepción (Chile)

Agradecimientos

A las personas que me acompañaron durante este proceso.

En especial a mi profesor guía Carlos Contreras por su paciencia y completa disposición a ayudar, el tiempo dedicado y los conocimientos entregados, y a toda mi familia y amigos por su apoyo incondicional y compañía durante esta etapa.

Powered@NLHPC: Este trabajo fue parcialmente apoyada por la infraestructura de supercómputo del NLHPC (ECM-02).

Resumen

La presente memoria de título tiene como objetivo presentar el problema de la ruta más corta múltiple con restricción de capacidades compartidas (SRMSPP), y usar una metaheurística para resolver dicho problema. Este problema fue recientemente publicado como una extensión del problema de ruta más corta con restricción de recursos (RCSPP). Esta variante presenta restricciones de consumo de recursos que son compartidos por arcos de una colección de grafos. Para resolver el SRMSPP se plantea un algoritmo de dos fases, la primera genera una solución inicial y la segunda busca mejorar la solución mediante una destrucción y reconstrucción de los grafos. Los resultados muestran la obtención de soluciones factibles en la gran mayoría de casos con tiempos elevados debido al tamaño y dificultad del problema.

Abstract

The objective of this thesis is to present the Shared Resource-Constrained Multi-Shortest Path Problem (SRMSPP), and use a metaheuristic to solve said problem. This problem was recently published as an extension of the Resource Constrained Shortest Path Problem (RCSPP). This variant introduces resource cosumption restrictions that are shared by arcs of a graph collection. To solve the SRMSPP, a two-phase algorithm is proposed, the first generates an initial solution and the second seeks to improve the solution by destroying and reconstructing the graphs. The results show the obtainment of feasible solutions in most cases with high times due to the size and difficult of the problem.

Tabla de contenido

Lista de Tablas

Lista de Figuras

Lista de Algoritmos

1 Introducción

En el presente capítulo se presenta una introducción al problema de ruta más corta y sus variantes. Además, los objetivos generales y específicos de la investigación y la estructura que presenta el documento.

1.1 Introducción

El problema de la ruta más corta es un típico problema en el área de investigación de operaciones, más específicamente en teoría de grafos. Su objetivo es encontrar el camino más corto entre dos vértices del grafo los cuales son llamados puntos de inicio y término. Esto no suele ser suficiente ya que muchas veces es necesario realizar la ruta más corta entre ciertos puntos específicos del grafo. Esto se puede dar en ciertos tipos de problemas como lo son el de ruteo de vehículos, donde se debe enviar mercancía a múltiples localizaciones designadas o el caso del problema del vendedor viajero donde la ruta debe pasar por todos los vértices del grafo. A esta clase de problemas se le denomina problema de planificación de ruta más corta multipunto o MPSP, por sus siglas en inglés (Multi-Point Shortest Path), y es considerado un problema *NP-Hard* (Liu, 2020).

El MPSP es más complejo debido a las restricciones que presentan los vértices, ya que la mayoría de los grafos son incompletos, por lo que para reducir la dificultad de resolverlo se realiza un método de conversión para transformarlo en un problema del vendedor viajero. Además, el MPSP es relevante y muy estudiado debido a su gran variedad de aplicaciones prácticas en diversos sectores como el turismo, transporte, redes eléctricas, reemplazo de equipos, diseño de robótica o telecomunicaciones entre otros (Obregón, 2005). Por ejemplo, en la planificación de atracciones turísticas se debe encontrar un camino que permita visitar todos los lugares de interés.

Para resolver este tipo de problemas han surgido una gran cantidad de algoritmos dependiendo de la cantidad de vértices que presente el grafo. Para el caso de pocos vértices, se han usado algoritmos clásicos que entregan soluciones exactas como branch and bound, branch and cut y programación dinámica (Liu, 2020). En cambio, cuando se tienen grafos con una mayor cantidad de vértices, se recurre a algunas metaheurísticas que entregan soluciones aproximadas tales como búsqueda tabú, algoritmos genéticos o de murciélago, entre otros (Liu, 2020).

Otra de sus variantes más populares es el problema de ruta más corta con restricción de recursos o RCSPP, por sus siglas en inglés (Resource Constrained Shortest Path Problem), donde el

consumo de recursos que están asociados a los arcos se convierte en una restricción adicional para el SPP original (Beasley, 1989). Una de sus extensiones menos estudiadas y que se aborda en este documento es el problema de la ruta más corta múltiple con restricción de capacidades compartidas o SRMSPP, por sus siglas en inglés (Shared Resource-Constrained Multi-Shortest Path Problem), donde el consumo de recursos ahora es compartido por arcos de varios grafos.

Entre las aplicaciones que tiene el SRMSPP se incluyen el control del tráfico aéreo, los problemas de reprogramación en trenes, servicios de recolección de basura además de problemas de programación de proyectos que involucren solo actividades en serie (García – Heredia, 2021).

Para el caso de este tipo de problemas debido a su naturaleza, los métodos usados en las otras variantes basados en la identificación y eliminación de rutas infactibles y dominadas no son aplicables. Es por esto que en la presente investigación se realiza una revisión de literatura del RCSPP y sus variantes para luego presentar una metaheurística que resuelva el SRMSPP. Esta consiste en la búsqueda inicial de una solución para luego realizar una destrucción y reconstrucción de grafos para mejorar la solución.

1.2 Objetivo general

Implementar una metaheurística para resolver el problema de la ruta más corta múltiple con restricción de capacidades compartidas.

1.3 Objetivos específicos

- Realizar la revisión de literatura de los métodos usados para resolver el SRMSPP.

- Diseñar e implementar una metaheurística para resolver el SRMSPP.

- Resolver las instancias del SRMSPP con la metaheurística.

- Analizar los resultados obtenidos y realizar una comparación con la literatura.

1.4 Estructura

El documento se organiza de la siguiente manera. En el Capítulo 2 se realiza una descripción detallada del SRMSPP junto a la formulación del modelo matemático y una revisión de literatura. Luego, en el Capítulo 3 se presenta una metaheurística, explicando la estructura general de esta y sus principales fases. El Capítulo 4 entrega los experimentos computacionales realizados, detallando su

implementación, describiendo las instancias utilizadas y la calibración de los parámetros, además de un análisis de los resultados obtenidos. Finalmente, en el Capítulo 5, se presentan las conclusiones obtenidas.

2 Problema de la ruta más corta múltiple con restricción de capacidades compartidas

En este capítulo se presenta una descripción del problema de la ruta más corta múltiple con restricción de capacidades compartidas. También, se presenta el modelo matemático como un problema de programación entera mixta. Finalmente, se hace una revisión de literatura para el RCSPP con sus principales metodologías de resolución.

2.1 Descripción del problema

El SRMSPP tiene como objetivo encontrar para cada grafo en el conjunto de grafos G_n, un camino entre un nodo de origen dado o_n y un nodo de destino dado d_n. Estos nodos pertenecen al conjunto de vértices V_n con n indicando el índice del grafo. Este camino o ruta debe utilizar el menor costo total sin exceder la capacidad W^r del conjunto de recursos R compartidos por los grafos, los cuales además presentan un consumo del recurso para cada arco denotado por w_a^r. Cada uno de los grafos presenta un conjunto de costos C_n asociados a cada arco del conjunto de arcos A_n, los cuales representan la entrada $\Lambda_n^+(i)$ o salida $\Lambda_n^-(i)$ del nodo i. Por simplicidad se utiliza la notación $(i,j)_n$ para representar un arco que va desde el nodo i al j en el grafo n.

A diferencia del RCSPP, en este problema los arcos de distintos grafos pueden compartir un mismo conjunto de recursos por lo que una ruta de un grafo puede influir directamente en la de otro. Esto conlleva a que si se toma la ruta de cada grafo de manera individual estos puedan ser factibles pero la solución completa no lo sea al considerar el consumo de recursos de todos los grafos en total.

Para ejemplificar lo anterior, en la Figura 2.1 se presenta un conjunto de tres grafos, cada uno con siete vértices con sus arcos y costos correspondientes, tomando como supuesto que cada arco consume el recurso asociado en su totalidad, es decir, con un $w_a^r = 1$. El objetivo del problema es encontrar la ruta mínima desde el nodo origen 1 hasta el de destino 7 para los tres grafos, pero considerando las restricciones de capacidad presentadas en Tabla 2.1.

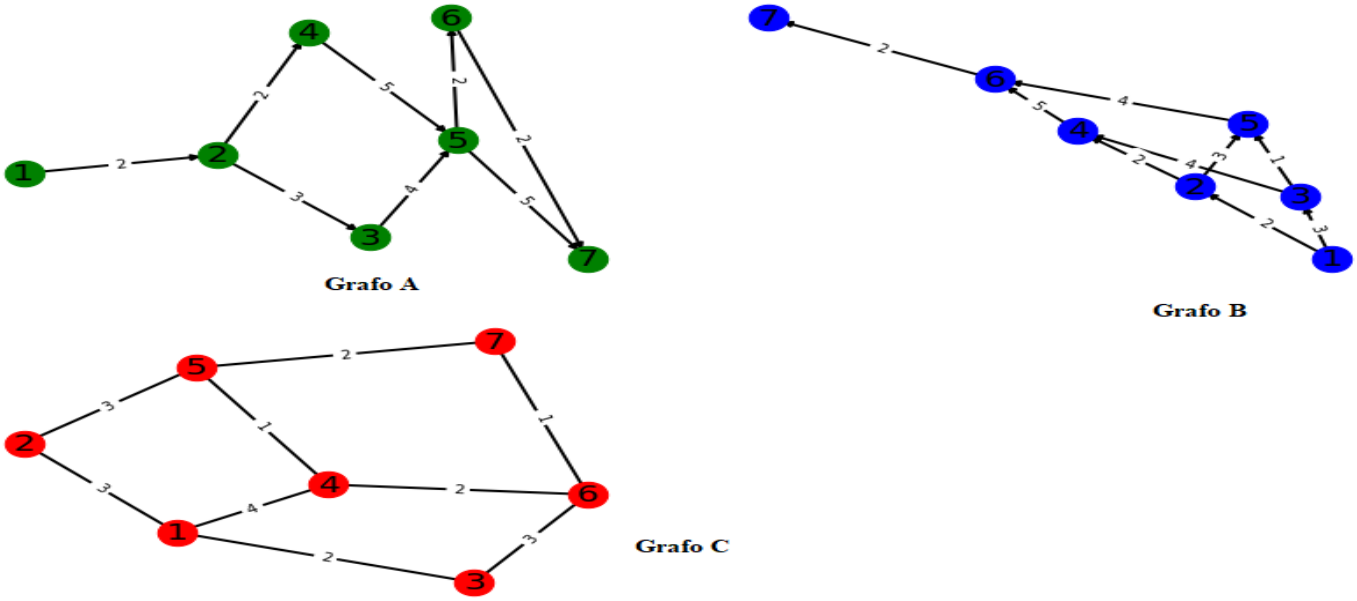

Figura 2.1: Representación grafos del problema ilustrativo.

Fuente: Elaboración propia.

Tabla 2.1: Recursos para el problema ilustrativo.

Recursos	Arcos Involucrados	Capacidad
R_1	$(2,4)_A, (3,5)_B, (3,6)_C$	2
R_2	$(5,6)_A, (4,6)_B, (4,5)_C$	1
R_3	$(3,5)_B, (2,5)_C$	1
R_4	$(4,5)_A, (5,7)_A, (2,4)_B, (6,7)_C$	3

Fuente: Elaboración propia.

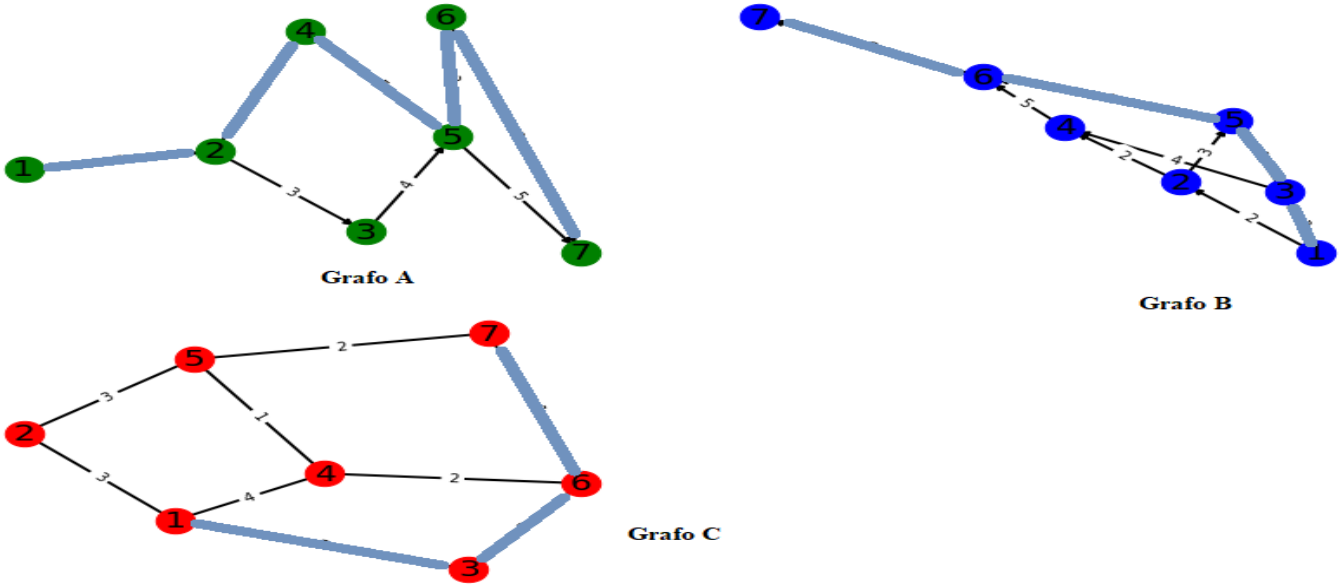

Figura 2.2: Representación ruta mínima del problema ilustrativo sin recursos compartidos.

Fuente: Elaboración propia.

La Figura 2.2 muestra la ruta mínima de cada grafo sin considerar capacidades de los recursos. Se observa que, en el SRMSPP, esta solución es infactible debido a que los arcos $(2,4)_A, (3,5)_B$ y $(3,6)_C$ forman parte de la solución total cuando la capacidad del recurso R_1 solo permite un máximo de 2 de estos arcos.

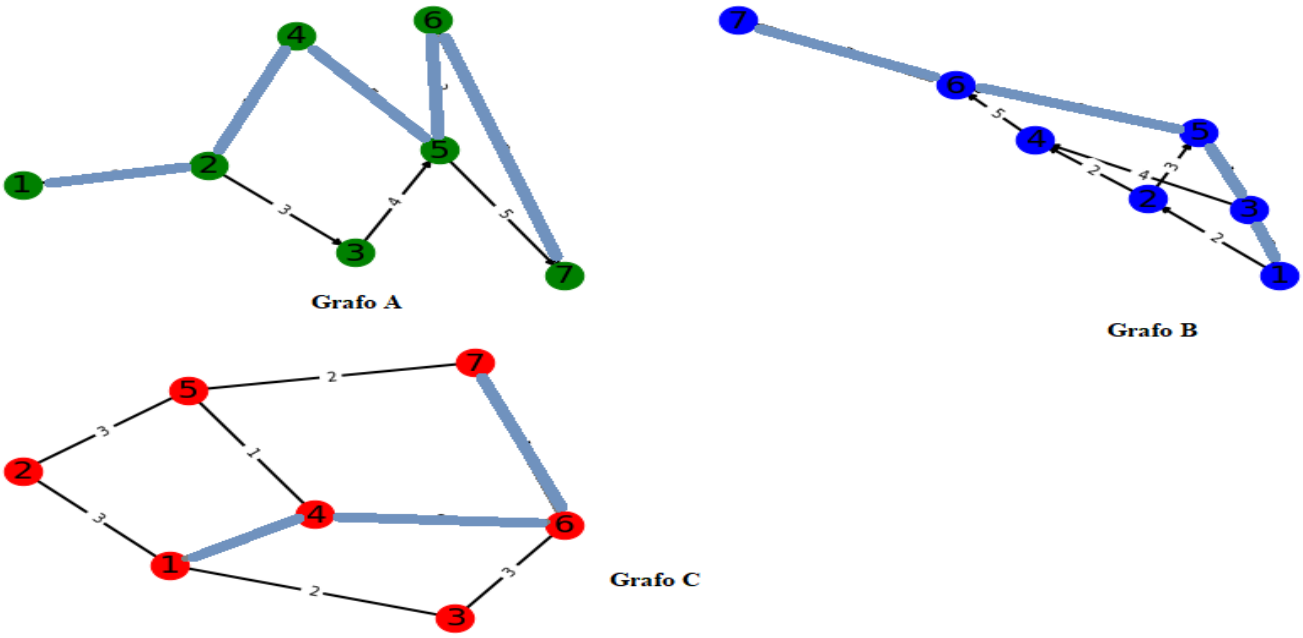

Figura 2.3: Representación ruta mínima del problema ilustrativo con recursos compartidos.

Fuente: Elaboración propia.

Considerando, restricciones de capacidad se tiene la solución de la Figura 2.3. Se observa que los grafos A y B presentan la misma ruta, y el grafo C presenta una variación con mayor costo a la anterior. Pero, si permite cumplir las capacidades establecidas evitando el arco $(3,6)_C$.

2.2 Modelo matemático

El SRMSPP se puede formular como un problema de programación entera (García – Heredia, 2021) considerando:

Variables de decisión:

x_a: $\begin{cases} 1, \text{ si el arco } a \in A_n, n \in N \text{ es parte de la solución.} \\ 0, \text{ en otro caso.} \end{cases}$

Parámetros:

c_a: Costo del arco a.

o_n: Nodo de origen del grafo n.

d_n: Nodo de destino del grafo n.

w_a^r: Consumo del recurso r por el arco a.

W^r: Capacidad del recurso r.

Modelo de programación matemática:

$$\min \sum_{n \in N} \sum_{a \in A_n} c_a x_a \qquad (1)$$

sujeto a:

$$\sum_{a \in \Lambda_n^-(i)} x_a - \sum_{a \in \Lambda_n^+(i)} x_a = \begin{cases} 1, & \text{si } i = o_n \\ 0, & \text{si } i = d_n \\ -1, & \text{en otro caso.} \end{cases} \qquad \forall i \in V_n, n \in N \qquad (2)$$

$$\sum_{n \in N} \sum_{a \in A_n} w_a^r x_a \leq W^r, \qquad \forall r \in R \qquad (3)$$

$$x_a \in \{0,1\}, \qquad \forall a \in A_n, n \in N \qquad (4)$$

La función objetivo (1) minimiza los costos de los arcos en la solución. Las ecuaciones (2) son restricciones de conservación de flujo para cada grafo, los arcos de cada grafo deben formar un camino

entre el nodo de origen y el de destino. El conjunto de restricciones (3) aseguran que se cumplan las capacidades de cada recurso y finalmente las restricciones (4) especifican que las variables son binarias.

2.3 Revisión de literatura

El RCSPP es una de las variantes más populares y por lo tanto, más estudiadas del SPP, lo que ha dado una gran cantidad de publicaciones con diversas variantes y métodos para su resolución. Dentro de estas, es posible destacar:

Handler et al. (1980) consideraron encontrar la ruta más corta entre dos nodos de un grafo sujeto a una restricción de mochila. Desarrollaron un algoritmo de relajación lagrangiana que busca reducir un valor k y así utilizar un k-ésimo algoritmo de ruta más corta, terminando con la primera ruta que satisface la restricción. Los resultados computacionales de este método indican órdenes de ahorros de magnitud cuando el enfoque se aplica a grafos de gran tamaño.

Beasley et al. (1989) presentan la problemática de un viajero con una cantidad de recursos determinada y que debe llegar a un destino lo más rápido posible dentro de las limitaciones impuestas por sus recursos. Para la resolución se desarrolla una relajación lagrangiana de la formulación de programación entera del problema de flujo de costo mínimo y así obtener un límite inferior para usarlo en un procedimiento de búsqueda de árbol. Esta metodología permite resolver problemas con tamaños de hasta 500 vértices, 5000 arcos y 10 recursos en menos de 30 segundos.

Debido a la escasez de comparaciones computacionales entre distintos algoritmos, Dumitrescu et al. (2003) realiza una comparación entre técnicas de escalada y un método estándar de establecimiento de etiquetas. Además, describe técnicas de preprocesamiento que aprovechan el costo y el límite superior a obtener de la información de la ruta más corta sin restricciones de recurso. Los experimentos computacionales demuestran que los preprocesamientos pueden llevar a mejoras importantes tanto en memoria requerida como en tiempo de ejecución.

Otras técnicas planteadas son las de programación lineal entera, donde Garcia (2009) estudia el politopo para el RCSPP, para el caso de un recurso obteniendo, nuevas clases de desigualdad validas. Además, se proporcionan rutinas de separación junto a un algoritmo de tiempo polinomial, para la construcción de grafos auxiliares. Además del desarrollo de un algoritmo de ramificación y corte junto a técnicas de preprocesamiento y ramificación que llevan a relajaciones de la programación

lineal y arboles de búsqueda equilibrados. Los resultados obtenidos muestran un mejor desempeño que los algoritmos de ramificación y corte predeterminados en ciertos softwares de programación.

Lozano et al. (2013) presentan un modelo de solución exacta para el problema de la ruta más corta con restricciones, capaz de manejar grafos de gran tamaño en un tiempo razonable. Este método consiste en un algoritmo que se basa en la idea de propagar pulsos a través del grafo, desde el nodo inicial al final. Construyendo rutas parciales que vayan cumpliendo los límites establecidos por las restricciones. Comparando los resultados con tres algoritmos diferentes se obtienen soluciones óptimas en grafos con tamaños cercanos a los 40.000 nodos y 800.000 arcos.

En busca de mejorar las metodologías ya planteadas, Horváth et al. (2016) investigan los métodos de ramificación basados en programación lineal. También, introduce nuevos planos de corte, procedimientos de separación, fijación de variables y métodos heurísticos primarios para resolver el RCSPP de manera óptima. Realizando los experimentos computacionales y la comparación con los otros métodos de la literatura, concluyen que los factores más importantes para resolver el RCSPP de manera eficiente con algoritmos de ramificación y poda, son: la fijación de variables y la heurística de búsqueda de la solución.

Este tipo de problemas también fue incorporado a otras variantes como son los subproblemas de ruteo de vehículos, el cual fue planteado por Chabrier (2006). Este estudio propone una mejora a los algoritmos de rutas elementales de este tipo de problemas, obteniendo mejores límites inferiores y haciendo una poda del árbol de búsqueda permitiendo encontrar soluciones exactas. Además, se muestra que este método se puede combinar con otras técnicas como la búsqueda local para mejorar los resultados.

Una extensión del problema anterior es el problema de ruteos de vehículos ecológicos, en los cuales los vehículos utilizan un combustible alternativo, el cual tienen capacidad limitada por lo que deben visitar estaciones de combustible. Montoya et al. (2016) propone una heurística de dos fases, creando un conjunto de rutas a través de heurísticas de rutas aleatorias, que insertan estaciones de combustible de forma óptima, para luego resolver una formulación de partición establecida sobres las rutas obtenidas anteriormente. Los resultados muestran que la heurística es competitiva con otros métodos de la literatura.

Debido a que este tipo de problemas se aplica en los campos del transporte y comunicación generalmente los parámetros de consumo de los recursos y los costos no se conocen con precisión. Di

Puglia (2019) modela esta variabilidad a través de un conjunto de incertidumbre definidos por otros autores. Para resolver el problema de manera óptima, se utiliza un enfoque de tres fases que calcula los límites robustos para la reducción de la dimensión del grafo. Además, usa la programación dinámica para reducir la brecha de dualidad. Los resultados obtenidos entregan una gran efectividad en instancias de menor tamaño, siendo más eficiente una combinación de lo planteado con otros algoritmos de literatura para problemas de mayor tamaño.

Otra variante estudiada últimamente es el problema de ruta multipunto, donde la ruta elegida como solución debe pasar por ciertos vértices dados. Liu et al. (2020) proponen una mejora a un tipo de algoritmos llamado de murciélago, que ya había sido usado por otros autores como Saji et al. (2015), para resolver el problema del vendedor viajero, pero que presentaba el inconveniente de converger demasiado pronto. Para solucionar esto plantean un operador de vecindario mejorado y además mejorar la estrategia de actualización de ubicaciones para operaciones de vecindario dinámicas. Los resultados indican que el algoritmo de murciélago formulado entrega mejores resultados que las alternativas presentes en la literatura en la mayoría de los casos.

Finalmente, García-Heredia et al. (2021) plantean una extensión del RCSPP, el cual no está presente en la literatura y que es abordado en este documento. El SRMSPP presenta restricciones de consumo de recursos, pero compartido por arcos de una colección de grafos. Los autores presentan una formulación de programación entera y un algoritmo de tres fases para resolver el problema. En primer lugar, una generación de soluciones para luego combinarlas en la segunda fase, y, por último, realizar una búsqueda local. Los resultados computacionales sugieren que el método propuesto es un enfoque eficaz para las diferentes variantes del problema de la ruta más corta.

3 Solución propuesta

En este capítulo se presenta el uso de una metaheurística para resolver el problema SRMSPP, realizando una explicación general del algoritmo junto a la representación de la solución. Finalmente, se describe en forma detallada cada fase de la metaheurística.

3.1 Estructura general

La metaheurística propuesta consta de dos fases, una de generación de una solución inicial y otra de destrucción y reconstrucción de grafos para mejorar la solución anterior. En la Sección 3.3 y 3.4, se proporcionan mayores detalles de las dos fases de la metaheurística. A continuación, en el Algoritmo 1 se presenta el pseudocódigo de la metaheurística propuesta.

Algoritmo 1: Metaheurística Propuesta

1. $s_{best}, f_{best}, \varepsilon = \text{SoluciónInicial}(G', G, W^r, C_i)$
2. **while** $i < I$
3. $G, G'' = \text{DestrucciónGrafos}(s_{best}, f_{best}, G, W^r, C_i, \alpha)$
4. $s, f, \varepsilon = SoluciónInicial(G'', G, W^r, C_i)$
5. **if** $\varepsilon = 1$
6. $\alpha = \alpha + 0.05$
7. **if** $\varepsilon = 0$ **and** $f < f_{best}$
8. $f_{best} = f; s_{best} = s$
9. $i = i + 1$
10. **return** s_{best}, f_{best}

Primero, se resuelve cada grafo buscando su ruta mínima con el algoritmo de Dijkstra (1959) y disminuyendo, los recursos asociados a cada arco de la solución. En el caso que una de las capacidades de un recurso llegue a 0, entonces se eliminan del conjunto de grafos, todos los arcos asociados a ese recurso. De esta manera no puedan tomar parte de la ruta mínima de los siguientes grafos a resolver. Si debido a la eliminación de arcos un grafo no encuentra camino, este se considera infactible y se pasa a resolver el siguiente (línea 1).

Luego de resolver la totalidad de los grafos y en busca de mejorar la solución obtenida se vuelve a resolver un porcentaje dado de estos grafos, tomando las soluciones infactibles y las que

presenten un costo mayor (líneas 3 y 4). Este proceso se realiza un numero dado de veces, aumentando el porcentaje si se encuentran rutas infactibles (líneas 5 y 6) y comparando siempre la solución total obtenida con la anterior, guardando la mejor y usándola como base para seguir iterando (líneas 7 y 8).

3.2 Representación de la solución

La solución obtenida por la metaheurística es representada como una lista de listas, donde se guardan las rutas más cortas para cada grafo. Por ejemplo, la solución de la Figura 2.3 del Capítulo 2, corresponde a la Ecuación (5).

$$s^f = \left[[1,2,4,5,6,7], [1,3,5,6,7], [1,4,6,7]\right] \qquad (5)$$

El primer elemento de la lista s^f corresponde a la ruta mínima en este caso del grafo A, el cual es una lista con los nodos que la forman. En caso de que uno de los grafos presente una solución infactible, en vez de la ruta encontrada, se indica la infactibilidad con un símbolo infinito ∞. Junto a la lista s^f, se tiene una lista f^f que indica el valor de la función objetivo para cada uno de los grafos, la cual es la suma de los costos de cada arco involucrado. Para el ejemplo de la Figura 2.3 del Capítulo 2 el f^f se muestra en la Ecuación (6).

$$f^f = [13, 10, 7] \qquad (6)$$

3.3 Solución inicial

Este procedimiento consiste en obtener una solución inicial para el SRMSPP, con el Algoritmo 2. Se recorre la lista G' que contiene todos los grafos a resolver en un orden aleatorio (línea 1). Luego, al grafo se le busca el camino mínimo con el algoritmo de Dijkstra y se retorna una lista con los vértices que forman la ruta llamada s y también el valor de la función objetivo f (línea 2). Posteriormente, se aplica una reducción de capacidad que toma la solución obtenida anteriormente junto a una lista W^r, guardando la capacidad que presenta cada recurso y otra llamada C_i con los costos asociados a cada arco. Esta reducción busca en la lista de capacidades, los recursos usados por cada arco de la solución y va descontando su valor. Si una de estas capacidades llega a 0, se trabaja con los arcos asociados a ese recurso para que no vuelvan a ser utilizados en los siguientes grafos. De manera, de ir aumentando el costo de los arcos en un valor tan grande que nunca sea tomado como parte de la solución por el algoritmo de Dijkstra. En el caso que no exista una ruta disponible o que no cumpla con las restricciones de capacidad, una variable llamada infactible toma el valor de 1, en

caso contrario, queda en 0 (línea 3). Luego de restar las capacidades, se consulta el valor de la variable infactible. Si su valor es 1, es decir, la solución es infactible, se ingresa un símbolo de infinito ∞ que indica que es infactible (líneas 4 y 5), además de una variable ε que indica si existe por lo menos una ruta infactible (línea 6). En caso contrario, la solución es factible y se guarda en una lista llamada s^0 (línea 8), para luego guardar el valor de la función objetivo a otra lista llamada f^0 (línea 9). Este proceso se repite para cada grafo de la lista G', retornando las listas s^0 y f^0.

Algoritmo 2: Solución Inicial

1. **for** i in G':
2. $s, f = \text{ResolverDijkstra}(G_i)$
3. infactible, W^r, $C_i = \text{ReducciónCapacidad}(s, W^r, C_i)$
4. **if** infactible $= 1$:
5. $s_i^0 = \infty$
6. $\varepsilon = 1$
7. **else**
8. $s_i^0 = s$
9. $f_i^0 = f$
10. **return** s^0, f^0, ε

3.4 Destrucción y reconstrucción

Para la segunda fase de este algoritmo, se realiza un proceso de destrucción y reconstrucción de ciertos grafos en busca de mejorar la solución obtenida inicialmente. Este se repite un número de iteraciones dado como parámetro. En el Algoritmo 3 se presenta la destrucción. Primero, se reciben las dos listas mencionadas con anterioridad y se devuelve otra lista G''. Ahora en G'', están ordenados los grafos, dejando en primer lugar los que tienen una solución infactible, para luego ordenar el resto de manera descendente acorde al valor de su función objetivo. Es decir, dejando al final de la lista los grafos que entreguen una función objetivo de menor valor (línea 1). Luego, a esta lista de grafos G'' se le retira un porcentaje α (ingresado como parámetro) (línea 2). Posteriormente, se le entrega la lista G'', los grafos G junto a sus costos C_i, devolviendo a su estado inicial los grafos presentes en la lista G'', y otorgándole a los arcos sus costos originales que pudieron ser modificados en la primera fase (línea 3). Además, las capacidades de los recursos W^r vuelven a sus valores iniciales, exceptuando

los arcos de las soluciones de los grafos que no fueron parte del porcentaje escogido en la etapa anterior (línea 4).

Algoritmo 3: Destrucción Grafos

1. $G'' = \text{OrdenarSolucionesCostosas}(s^0, f^0)$
2. $G'' = \text{Porcentaje}(G'', \alpha)$
3. $G = \text{RestaurarCostos}(G, G'', C_i)$
4. $G = \text{RestaurarCapacidades}(G, G'', W^r)$
5. **return** G, G''

Luego, ocurre la reconstrucción de los grafos, donde la lista de grafos G'' se resuelve nuevamente siguiendo los pasos explicados en la fase de solución inicial y presentados en el Capítulo 3.3. Con las nuevas rutas obtenidas, en el Algoritmo 1 tras cada iteración, si existe la presencia de soluciones infactibles, el porcentaje α aumenta en un 5% (líneas 5 y 6). Luego de cada iteración, si las soluciones s son factibles en su totalidad y si el valor de la función objetivo f es menor a la mejor solución obtenida con anterioridad, llamada f_{best}, la nueva solución s pasa a ser s_{best}, ocurriendo lo mismo con el valor de la función f. En caso contrario, la solución no cambia y se vuelve a iterar hasta I, entregando la lista s_{best} con la mejor combinación de rutas mínimas encontradas en el número de iteraciones y su respectivo valor f_{best} (líneas 7 y 8).

4 Experimentos computacionales

Para analizar el rendimiento de la metaheurística planteada, se realizaron diversos experimentos para los cuales se utiliza las instancias generadas por García-Heredia et al. (2021) basadas en vuelos domésticos de Estados Unidos. Estos experimentos consisten en resolver 10 veces cada instancia usando una semilla diferente para así obtener un promedio del valor de la función y del tiempo de cada una.

La implementación de la metaheurística fue hecha en el lenguaje de programación Python 3 y ejecutada en un computador con procesador 2 x Intel Xeon Gold 6152 CPU @ 2.10GHz, 22 cores C./U, memoria RAM de 768 GB y usando el sistema operativo CentOS Linux 7. Todos los experimentos fueron ejecutados usando solo un hilo del procesador (secuencialmente).

4.1 Descripción de las instancias

Las instancias propuestas por García-Heredia et al. (2021), vienen de datos de vuelos hechos en Estados Unidos el día 16 de enero, mayo y septiembre del año 2019. Se elige este día debido a que es el que presenta un mayor volumen de tráfico aéreo en cada uno de los meses. Para tener instancias con distintos tamaños, los autores crearon versiones más pequeñas basadas en los datos originales obteniendo tres conjuntos de datos, uno con un 30% del total de aviones, otro con un 65% de estos y finalmente un conjunto con la totalidad de los aviones presente en los datos. Las dimensiones de estas instancias se presentan en la Tabla 4.1. Para cada uno de los casos de la tabla, hay 12 casos basados en niveles de capacidad, los cuales se agrupan en tres categorías (fácil, medio y difícil). Estos fueron generados de la siguiente manera: primero se crea un escenario base con valores de capacidad, que se basan en el mínimo necesario para que sea factible la instancia de vuelo original. En cada categoría, se tienen cuatros escenarios dependiendo de la reducción de capacidad, la cual puede ser del $10\%, 20\%, 30\%$ *y* 40%. Finalmente, para dos categorías se aplica una reducción adicional dependiendo de su nivel de dificultad, en el caso de la categoría medio se aplica una reducción de capacidad aleatoria al 50% de los elementos que puede tomar un valor entre un 1% y un 20%. Para el caso de la categoría difícil, esta reducción aleatoria de capacidad se aplica a todos los elementos.

Tabla 4.1: Dimensiones de las instancias.

Mes	Tamaño (%)	Vuelos	Grafos	Arcos	Nodos	Recursos
Enero	30	5596	1329	3116931	1482732	136887
Mayo	30	6951	1368	4043046	1868888	145711
Septiembre	30	6610	1368	3670644	1729221	142734
Enero	65	11788	2879	7195904	3336779	173249
Mayo	65	13752	2963	9717333	4341805	178451
Septiembre	65	13530	2964	8597363	3922250	185519
Enero	100	18100	4429	11934419	5446911	184404
Mayo	100	20634	4558	14475487	6457815	204724
Septiembre	100	20581	4559	13993972	6248302	201451

Fuente: Elaboración propia.

En resumen, se tienen datos para tres meses, cada uno de estos presenta tres instancias de diferentes tamaños, se les aplica una reducción de capacidad separada en tres categorías y con cuatro escenarios posibles para cada uno, formando un total de 108 instancias para probar la metaheurística.

4.2 Calibración de parámetros

La metaheurística propuesta presenta solo dos parámetros, α y I. α corresponde a un valor entre 0 y 1, el cual indica el porcentaje de grafos que se vuelven a resolver. El parámetro I indica el número de iteraciones que realiza la metaheurística para encontrar una mejor solución. Para encontrar los valores de los parámetros que entreguen mejores soluciones, se realiza un proceso de calibración, que consiste en ejecutar reiteradas veces la metaheurística utilizando un número establecido de instancias, modificando los parámetros en un determinado rango con el fin de encontrar el valor adecuado. Los resultados obtenidos se presentan en la Tabla 4.2 junto al rango en el que fueron evaluados.

Tabla 4.2: Resultados de la calibración de los parámetros.

Parámetro	**Rango**	**Valor**
α	$[75, 80, 85, 90, 95]$	90
I	$[5, 10, 15, 20, 25]$	20

Fuente: Elaboración propia.

Para la elección del valor, se analiza las mejores soluciones promedio obtenidas que se presentan en la Tabla 4.3. Se observa que las mejores soluciones son entregadas cuando el parámetro α toma el valor de 90. En cuanto al parámetro I, ya que es el número de iteraciones, si este valor aumenta las probabilidades de encontrar una mejor solución también, lo cual indicaría que el valor a tomar debería ser 25. Al revisar las soluciones obtenidas junto al tiempo de ejecución, el cual es presentado en la Tabla 4.4, se tiene que no es recomendable tomar este valor de 25, debido a que la mejora de la solución es mínima a diferencia del tiempo de ejecución que aumenta considerablemente.

Tabla 4.3: Mejor solución promedio para cada combinación de parámetros.

α/I	5	10	15	20	25
75	2277.21	2232.64	2330.57	2191.99	2191.30
80	2258.45	2203.02	2183.31	2161.77	2153.37
85	2239.56	2180.13	2172.21	2167.53	2137.22
90	2158.91	2131.97	2125.70	2103.23	2100.85
95	2206.88	2148.10	2142.60	2141.88	2137.31

Fuente: Elaboración propia.

Tabla 4.4: Tiempo de ejecución promedio (segundos) para cada combinación de parámetros.

α/I	5	10	15	20	25
75	155.92	273.71	370.36	453.42	530.98
80	162.55	283.64	378.85	465.87	548.27
85	166.65	291.18	391.40	482.11	566.63
90	171.97	304.30	404.05	498.57	590.90
95	175.30	310.52	418.64	510.61	607.04

Fuente: Elaboración propia.

4.3 Resultados

Al ejecutar las 108 instancias no se encontraron soluciones factibles en dos de estas, las cuales corresponden al día 16 de septiembre, con nivel de capacidad difícil y reducción de capacidad del 40% para un tamaño del 65% y del 100%. Además, en 11 instancias se encontró a lo menos una solución infactible dentro de las 10 ejecuciones realizadas. Comparando con los resultados obtenidos por García-Heredia et al. (2021), en este se encuentran 44 instancias que no pueden ser resueltas debido a falta de memoria sin encontrar una solución entera. En la Tabla 4.5 se presentan los resultados obtenidos para las instancias del mes de enero con un tamaño del 30% de los datos originales, indicando el mejor costo obtenido junto al costo y tiempo promedio de las 10 ejecuciones realizadas. Para observar los resultados obtenidos de las otras instancias revisar Anexo.

Tabla 4.5: Resultados obtenidos para las instancias del mes de enero con un tamaño del 30%.

Dificultad	**Reducción Capacidad (%)**	**Mejor Costo**	**Costo Promedio**	**Tiempo Promedio**
Fácil	10	64.69	80.04	410.52
Fácil	20	226.76	259.26	384.42
Fácil	30	640.39	701.67	377.9
Fácil	40	1537.57	1774.48	353.71
Medio	10	3165.73	3612.73	437.02
Medio	20	3647.02	4137.39	435.95

Medio	30	4620.74	5220.37	379.49
Medio	40	7032.27	7840.28	381.98
Difícil	10	5500.96	6537.44	379.37
Difícil	20	6064.5	6659.9	380.14
Difícil	30	6248.79	7335.96	370.6
Difícil	40	6573.57	Infactible	366.06

Fuente: Elaboración propia.

En cuanto a los tiempos obtenidos en promedio, se presenta en la Tabla 4.6 con los resultados para cada mes con tamaños de 30% y 65% además de los obtenidos por García-Heredia et al. (2021). Si se comparan estos tiempos se puede apreciar que generalmente la metaheurística presenta tiempos de ejecución elevados y que siendo comparados con la literatura solo suelen ser menores en las instancias con dificultad difícil. Se debe tener en cuenta como se menciona en la Sección 4.2 que una reducción en el número de iteraciones reduciría los tiempos, pero podría afectar la calidad de la solución obtenida. Además, el lenguaje de programación influye en estos tiempos de ejecución, ya que Python 3 es aproximadamente 46 veces más lento que C++ (Pereira, 2017), el cual fue utilizado por García-Heredia et al. (2021). Si los tiempos obtenidos se ajustan a la relación anteriormente mencionada entre los lenguajes de programación, se tienen los tiempos de ejecución de la Tabla 4.7 como referencia. Se debe tener en cuenta que esta comparación entre los tiempos de los lenguajes de programación es obtenida de un promedio al ejecutar 10 problemas diferentes de programación por Pereira et al. (2017) y que no necesariamente son representativos para este caso debido a los diversos factores involucrados como puede ser el hardware, tipo de problema, optimización del código entre otros. Pero, si pueden servir como una buena referencia de la diferencia de tiempo que se puede conseguir al programar la metaheurística en otro lenguaje de programación.

Tabla 4.6: Comparación entre tiempos de ejecución promedio (segundos).

Mes	**Tamaño (%)**	**Dificultad**	**Tiempo Promedio**	
			Metaheurística	**García-Heredia et al.**
Enero	30	Fácil	381.64	124.84
Enero	30	Medio	408.61	238.34
Enero	30	Difícil	374.04	337.61
Mayo	30	Fácil	494.48	172.56
Mayo	30	Medio	521.94	253.61
Mayo	30	Difícil	510.81	722.91
Septiembre	30	Fácil	434.43	155.68
Septiembre	30	Medio	458.42	163.59
Septiembre	30	Difícil	419.90	505.21
Enero	65	Fácil	895.12	335.97
Enero	65	Medio	955.98	856.33
Enero	65	Difícil	848.35	2117.18
Mayo	65	Fácil	1235.49	523.56
Mayo	65	Medio	1306.38	1271.99
Septiembre	65	Fácil	1084.50	374.24
Septiembre	65	Medio	1142.42	862.39

Fuente: Elaboración propia.

Tabla 4.7: Tiempos de ejecución promedio (segundos) siguiendo la equivalencia entre Python y C++.

Mes	Tamaño (%)	Dificultad	Tiempo Promedio
Enero	30	Fácil	8.30
Enero	30	Medio	8.88
Enero	30	Difícil	8.13
Mayo	30	Fácil	10.75
Mayo	30	Medio	11.35
Mayo	30	Difícil	11.10
Septiembre	30	Fácil	9.44
Septiembre	30	Medio	9.97
Septiembre	30	Difícil	9.13
Enero	65	Fácil	19.46
Enero	65	Medio	20.78
Enero	65	Difícil	18.44
Mayo	65	Fácil	26.86
Mayo	65	Medio	28.40
Septiembre	65	Fácil	23.58
Septiembre	65	Medio	24.84

Fuente: Elaboración propia.

5 Conclusiones

En este trabajo de titulación se presenta una metaheurística basada en una estrategia de destrucción y reconstrucción para resolver el SRMSPP. Esta consta de dos fases y dos parámetros para su ejecución, entregando soluciones factibles en la gran mayoría de instancias, incluidas las de mayor tamaño y dificultad. En cuanto a los tiempos, estos son por lo general mayores que los vistos en la literatura exceptuando la dificultad más alta.

Debido a la dificultad del problema y al gran tamaño de las instancias es necesario gran capacidad de memoria y tiempos elevados para encontrar soluciones, esto se aprecia en el algoritmo propuesto y también en los resultados que se compararon de la literatura donde gran parte de las instancias no obtienen solución debido a la falta de memoria a y los tiempos límites impuestos.

Como este problema es relativamente nuevo, se espera que surjan nuevas investigaciones a futuros, con nuevos algoritmos para resolver el SRMPSPP o mejoras a los métodos ya propuestos, ya sea en busca de reducir el tiempo necesario para obtener buenas soluciones o para mejorar la calidad de estas. En esta última dirección como trabajo futuro se puede mejorar el algoritmo implementándolo en C++ y también realizando una destrucción más fina del grafo y no completa.

6 Referencias

Beasley, J. E. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, 19, 379–394.

Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33, 2972–2990.

Di Puglia Pugliese, L., Guerriero, F., & Poss, M. (2019). The Resource Constrained Shortest Path Problem with uncertain data: A robust formulation and optimal solution approach. *Computers & Operations Research*, 107, 140-155.

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik 1*, 269-271.

Dumitrescu, I., & Boland, N. (2003). Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks: An. International Journal*, 42, 135–153.

Garcia, R. (2009). Resource constrained shortest paths and extensions. *Ph.D. thesis Georgia Institute of Technology*.

García-Heredia, D., Molina, E., Laguna, M., & Alonso-Ayuso, A. (2021). A solution method for the shared resource-constrained multi-shortest path problem. *Expert Systems With Applications*, 182, 115193.

Handler, G. Y., & Zang, I. (1980). A dual algorithm for the constrained shortest path problem. *Networks, 10*, 293–309.

Horváth, M., & Kis, T. (2016). Solving resource constrained shortest path problems with lp-based methods. *Computers & Operations Research*, 73, 150–164.

Liu, L., Luo, S., Guo, F., & Tan, S. (2020). Multi-point shortest path planning based on an Improved Discrete Bat Algorithm. *Applied Soft Computing Journal*, 1.

Lozano, L., & Medaglia, A. L. (2013). On an exact method for the constrained shortest path problem. *Computers & Operations Research*, 40, 378–384.

Montoya, A., Gu´eret, C., Mendoza, J. E., & Villegas, J. G. (2016). A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, 70, 113–128.

Obregón Quintana, B. (2005). Teoría de Redes: El Problema de la Ruta más Corta. *Programa de Maestría y Doctorado en Ingeniería*.

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2017). Energy efficiency across programming languages: how do energy, time, and memory relate? *SLE 2017: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, 256-267.

Saji, Y., & Riffi, M. (2015). A novel discrete bat algorithm for solving the travelling salesman problem. *Neural Comput. Appl*, 27, 1-14.

7 Anexos

7.1 Resultados obtenidos en el resto de las instancias

Tabla 7.1: Resultados obtenidos para las instancias del mes de mayo con un tamaño del 30%.

Dificultad	Reducción Capacidad (%)	Mejor Costo	Costo Promedio	Tiempo Promedio
Fácil	10	88.67	98.08	522.87
Fácil	20	355.57	394.97	524.14
Fácil	30	1281.61	1375.29	464.76
Fácil	40	3544.35	3776.86	466.14
Medio	10	3354.34	3526.96	550.8
Medio	20	4070.65	4347.71	553.25
Medio	30	5329.69	5524.41	492.89
Medio	40	7546.47	8105.77	490.83
Difícil	10	7637.96	8176.94	496.95
Difícil	20	7984.07	8861.29	496.63
Difícil	30	8695.01	9538.27	525.68
Difícil	40	11168.92	11933.65	523.98

Fuente: Elaboración propia.

Tabla 7.2: Resultados obtenidos para las instancias del mes de septiembre con un tamaño del 30%.

Dificultad	**Reducción Capacidad (%)**	**Mejor Costo**	**Costo Promedio**	**Tiempo Promedio**
Fácil	10	129.53	146.46	461.91
Fácil	20	262.8	283.91	464.97
Fácil	30	771.26	847.99	405.8
Fácil	40	2201.41	2460.84	405.04
Medio	10	2155.31	2193.07	487.27
Medio	20	2267.5	2363.66	489.9
Medio	30	2653.15	2795.21	429.31
Medio	40	3499.44	3746.16	427.2
Difícil	10	6626.18	7351.02	425.84
Difícil	20	7651.62	8169.51	418.29
Difícil	30	8807.86	9591.85	444.3
Difícil	40	12507.35	13295.69	391.18

Fuente: Elaboración propia.

Tabla 7.3: Resultados obtenidos para las instancias del mes de enero con un tamaño del 65%.

Dificultad	**Reducción Capacidad (%)**	**Mejor Costo**	**Costo Promedio**	**Tiempo Promedio**
Fácil	10	75.2	81.55	926.75
Fácil	20	219.34	240.65	879.07
Fácil	30	1058.03	1165.61	880.87
Fácil	40	3713.02	3893.9	893.78
Medio	10	5762.94	6023.99	990.27
Medio	20	6048.88	6291.98	942.71
Medio	30	7658.99	8090.92	944.08
Medio	40	12435.97	Infactible	946.87
Difícil	10	12073.68	12579.72	900.67
Difícil	20	12911.21	13987.95	773.51
Difícil	30	14040.03	Infactible	872.94
Difícil	40	17183.95	Infactible	846.27

Fuente: Elaboración propia.

Tabla 7.4: Resultados obtenidos para las instancias del mes de mayo con un tamaño del 65%.

Dificultad	**Reducción Capacidad (%)**	**Mejor Costo**	**Costo Promedio**	**Tiempo Promedio**
Fácil	10	198.45	230.61	1269.77
Fácil	20	839.52	992.41	1225.86
Fácil	30	2796.5	2917.24	1232.79
Fácil	40	7227.79	7751.41	1213.54
Medio	10	5627.47	5868.75	1321.78
Medio	20	6914.72	7456.63	1294.02
Medio	30	12100.57	12739.75	1322.3
Medio	40	21793.47	23638.43	1287.42
Difícil	10	10893.21	11937.1	1197.37
Difícil	20	12287	13153.87	1172.55
Difícil	30	15478.43	16783.9	1264.35
Difícil	40	24290.91	Infactible	1289.11

Fuente: Elaboración propia.

Tabla 7.5: Resultados obtenidos para las instancias del mes de septiembre con un tamaño del 65%.

Dificultad	**Reducción Capacidad (%)**	**Mejor Costo**	**Costo Promedio**	**Tiempo Promedio**
Fácil	10	54.42	60.58	1127.26
Fácil	20	72.75	78.06	1065.88
Fácil	30	96.26	111.44	1074.39
Fácil	40	164.57	176.91	1070.46
Medio	10	6015.83	6879.66	1178.48
Medio	20	6989.33	7371.27	1137.86
Medio	30	7772.57	8167.31	1130.68
Medio	40	10980.38	Infactible	1122.64
Difícil	10	13324.23	Infactible	1105.08
Difícil	20	15908.35	Infactible	1115.2
Difícil	30	23172.15	Infactible	1118.29
Difícil	40	Infactible	Infactible	1118.82

Fuente: Elaboración propia.

Tabla 7.6: Resultados obtenidos para las instancias del mes de enero con un tamaño del 100%.

Dificultad	**Reducción Capacidad (%)**	**Mejor Costo**	**Costo Promedio**	**Tiempo Promedio**
Fácil	10	54.73	69.73	1577.24
Fácil	20	67.25	76.89	1518.16
Fácil	30	150.17	162.09	1502.85
Fácil	40	222.02	264.7	1376.96
Medio	10	7174.52	7530.29	1679.06
Medio	20	6887.32	7376.85	1636.66
Medio	30	7151.55	7667.81	1597.41
Medio	40	7153.1	7704.4	1428.58
Difícil	10	16534.46	17567.5	1575.78
Difícil	20	17523.75	18510.85	1593.71
Difícil	30	21585.05	21994.39	1555.78
Difícil	40	28315.72	Infactible	1596.14

Fuente: Elaboración propia.

Tabla 7.7: Resultados obtenidos para las instancias del mes de mayo con un tamaño del 100%.

Dificultad	**Reducción Capacidad (%)**	**Mejor Costo**	**Costo Promedio**	**Tiempo Promedio**
Fácil	10	262.68	311.85	1865.23
Fácil	20	1235.59	1332.89	1867.06
Fácil	30	4003.18	4414.63	1867.84
Fácil	40	11357.05	12242.34	1878.47
Medio	10	5208.59	5444.47	1934.57
Medio	20	6646.02	6853.99	1928.75
Medio	30	9993.88	10737.37	1935.8
Medio	40	19177.81	20276.73	1896.38
Difícil	10	13374	13663.3	1776.17
Difícil	20	13497.5	14287.16	1747.84
Difícil	30	15004.4	16044.31	1744.21
Difícil	40	21081.57	22191.66	1732.56

Fuente: Elaboración propia.

Tabla 7.8: Resultados obtenidos para las instancias del mes de septiembre con un tamaño del 100%.

Dificultad	**Reducción Capacidad (%)**	**Mejor Costo**	**Costo Promedio**	**Tiempo Promedio**
Fácil	10	173.75	189.11	1752.85
Fácil	20	855.04	941.8	1747.97
Fácil	30	3160.6	3428.62	1760.43
Fácil	40	11371.81	11982.49	1806.73
Medio	10	7040.66	7450.85	1858.22
Medio	20	8149.67	8477.66	1876.13
Medio	30	12051.59	12548.81	1878.02
Medio	40	21307.92	23615.04	1858.67
Difícil	10	17283.81	18925.88	1711.85
Difícil	20	19844.23	21605.6	1693.85
Difícil	30	27529.14	Infactible	1700.81
Difícil	40	Infactible	Infactible	1721.95

Fuente: Elaboración propia.

UNIVERSIDAD DE CONCEPCION – FACULTAD DE INGENIERIA
RESUMEN DE MEMORIA DE TITULO

Departamento de Ingeniería		Industrial	
Título		Una metaheurística para el problema de planificación de la ruta más corta multipunto	
Nombre Memorista		Esteban Vallejos Yévenes	
Modalidad	Investigación	Profesor(es) Patrocinante	
Concepto		Carlos Contreras Bolton	
Calificación			
Fecha		Ingeniero Supervisor	Institución
Comisión (Nombre y Firma)			
Resumen			

La presente memoria de título tiene como objetivo presentar el problema de la ruta más corta múltiple con restricción de capacidades compartidas (SRMSPP), y usar una metaheurística para resolver dicho problema. Este problema fue recientemente publicado como una extensión del problema de ruta más corta con restricción de recursos (RCSPP). Esta variante presenta restricciones de consumo de recursos que son compartidos por arcos de una colección de grafos. Para resolver el SRMSPP se plantea un algoritmo de dos fases, la primera genera una solución inicial y la segunda busca mejorar la solución mediante una destrucción y reconstrucción de los grafos. Los resultados muestran la obtención de soluciones factibles en la gran mayoría de casos con tiempos elevados debido al tamaño y dificultad del problema.