UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

# UN MODELO Y ALGORITMO PARA EL PROBLEMA FLEXIBLE DEL VENDEDOR VIAJERO CON MÚLTIPLES DRONES

**Por: Benjamín Brandt Mieres**

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

Agosto 2023
Concepción, Chile

**Profesor Guía: Carlos Contreras Bolton**

© 2023, Benjamín Brandt Mieres
Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento

# AGRADECIMIENTOS

En primer lugar quiero agradecer a mis padres y hermanos por su constante apoyo a lo largo de mi vida. Sus consejos y aliento han sido fundamentales en mi desarrollo. También, agradecer a mis amigos por compartir risas y alegrías en cada paso del camino, haciendo que incluso los momentos más desafiantes se conviertan en buenos recuerdos. Finalmente, agradecer a mis profesores por su dedicación en la enseñanza y brindarme herramientas valiosas para crecer.

Esta memoria de título fue parcialmente apoyada por la infraestructura de supercómputo del NLHPC (ECM-02).

# Resumen

En el último tiempo han existido desarrollos y aplicaciones importantes en la logística. Diferentes investigaciones demuestran que el uso de drones como herramienta para la entrega de paquetes trae beneficios significativos. Esta memoria de título aborda el problema flexible del vendedor viajero con múltiples drones (Flexible Drones Traveling Salesman Problem, FDTSP por sus siglas en inglés). Este problema busca la combinación óptima que minimice el tiempo para realizar entregas entre un camión y múltiples drones. Este tipo de problema se caracteriza por su complejidad computacional al ser abordado, es por ello que los algoritmos de la literatura obtienen soluciones que a veces se alejan de los óptimos. En el presente estudio se propone un modelo de programación lineal entera mixta y un algoritmo variable neighborhood search (VNS). El VNS propuesto opera en dos fases, una fase enfocada en optimizar la ruta del camión y la segunda fase está centrada en optimizar las rutas de los drones. La primera fase optimiza mediante operadores basados en el problema del vendedor viajero. Posteriormente, en la segunda fase se utiliza un algoritmo simulated annealing para mejorar las rutas de los drones. El modelo y algoritmo son validados en dos conjuntos de instancias que representan condiciones de operación en entornos urbanos, suburbanos y rurales. Los resultados computacionales muestran que el algoritmo propuesto es mejor que los algoritmos de la literatura en la mayoría de las instancias evaluadas, respaldado por un análisis estadístico. Sin embargo, el rendimiento empeora a medida que el número de drones aumenta. En tanto, el modelo logra garantizar la optimalidad en instancias de hasta ocho vértices, evidenciando que para instancias de 20 o más vértices se debe usar una metaheurísitica como el VNS propuesto.

***Keywords*** – Logística, Drones, FDTSP, Programación Lineal Entera-Mixta, Variable Neighborhood Search

# Abstract

In recent times, there have been significant developments and applications in logistics. Various studies demonstrate that the use of drones as a tool for package delivery brings significant benefits. This thesis addresses the Flexible Drones Traveling Salesman Problem (FDTSP), which seeks to find the optimal combination of a truck and multiple drones to carry out a route delivering packages to customers in minimum time. This type of problem is characterized by its computational complexity to be addressed. That is why algorithms from the literature sometimes obtain solutions that deviate from the optima. In the present study, we propose an integer mixed linear programming model and a variable neighborhood search algorithm (VNS). The proposed VNS optimizes a solution in two phases, with the first phase focused on optimizing the truck's route and the second phase centered on optimizing the drone routes. The initial phase optimizes using operators based on the Traveling Salesman Problem. Subsequently, in the second phase, a simulated annealing algorithm is employed to enhance the drone routes. The model and VNS are tested on two sets of instances representing operational conditions in urban, suburban, and rural environments. Computational results show that the proposed algorithm outperforms the algorithms from the literature in the majority of the evaluated instances, supported by statistical analysis. However, performance deteriorates as the number of drones increases. Meanwhile, the model succeeds in ensuring optimality in instances with up to eight vertices, demonstrating that for instances with 20 or more vertices, a metaheuristic should be employed as the proposed VNS.

***Keywords*** – Logistics, Drones, FDTSP, Mixed-Integer Linear Programming, Variable Neighborhood Search

# Índice general

# Índice de Tablas

# Índice de Figuras

# Capítulo 1

# Introducción

En este capítulo se presenta una breve motivación e introducción del trabajo, el objetivo general y los objetivos específicos.

## 1.1. Motivación

Desde comienzos de la pasada década, se ha incrementado considerablemente el uso de vehículos aéreos no tripulados como herramienta de apoyo para realizar tareas de distintas complejidades. Gracias a sus capacidades, potencialidades y el uso de tecnologías de información, han surgido diversas aplicaciones para estos dispositivos en distintos rubros.

Algunas de las tantas aplicaciones incluyen la agricultura, construcción, emergencias, entre otras. Así en la agricultura, los drones pueden ser utilizados para mapear y organizar agricultura de precisión, clasificación y vigilancia de cultivos, aplicación de pesticidas y fertilizantes, conservación de la fauna y flora, administración del uso de agua y medición de estrés hídrico, detección de malezas, pestes o enfermedades (Rejeb *et al.*, 2022). En la construcción pueden ser usados como herramientas de inspección, mapeo, mantención y monitoreo de procesos y seguridad. Reduciendo costos y mejorando el desempeño de las actividades y la relación con las personas implicadas en el rubro (Nwaogu *et al.*, 2023). En contextos de emergencias relacionadas a desastres naturales, estos vehículos pueden ser ocupados para el mapeo y organización de terrenos, búsqueda y rescate, transporte y entrenamiento (Mohd Daud *et al.*, 2022). Además, se pueden utilizar en la prestación de servicios médicos, ya sea en emergencia o no, facilitando el envío y entrega de medicamentos

e insumos médicos (Roberts *et al.*, 2023). De igual forma, los drones se pueden desempeñar en tareas relacionadas con la logística. Utilizados por empresas distribuidoras para mejorar la eficiencia de los servicios de entregas, como respuesta de los servicios de reparto a la creciente demanda proveniente de servicios relacionados al comercio en línea (Guthrie *et al.*, 2021). De esta manera se reducen costos, tiempo o distancias recorridas, con los drones como complementos o incluso reemplazando a otro tipo de vehículos para realizar entregas (Ha *et al.*, 2018) y abordar de manera más eficaz el denominado problema de la *última milla*.

A pesar de que tradicionalmente el problema de la *última milla* ha sido tratado mediante el planteamiento del problema del vendedor viajante (TSP por sus siglas en inglés: *Traveling Salesman Problem*), existen variados enfoques para minimizar los costos en un sistema de reparto. Uno de estos es el enfoque de ayudar a la flota de camiones con el uso de tecnología de drones. Murray & Chu (2015) realizaron uno de los primeros trabajos que utilizaron un enfoque de FSTSP (por sus siglas en inglés: *Flying Sidekick Traveling Salesman Problem*). Luego, Agatz *et al.* (2018) abordan el TSP-D (por sus siglas en inglés: *Traveling Salesman Problem with Drone*). A través de los años, distintos autores han abordado diversamente el problema estableciendo supuestos dado los contextos establecidos, uno de estos definido por Cavani *et al.* (2021), el TSP-MD (por sus siglas en inglés: *Traveling Salesman Problem with Multiple Drones*). Sin ir más allá, en Lu *et al.* (2022) motivados por el TSP-D proponen distintos métodos de solución para la expansión del TSP-D con múltiples drones. Lo llaman FDTSP (por sus siglas en inglés: *Flexible Drone Traveling Salesman Problem*) por los métodos de solución que utilizan y la introducción de ciertos supuestos que lo diferencian del TSP-MD.

El FDTSP es una variante del TSP-MD para resolver el problema mediante la asignación de rutas para un único camión y uno o múltiples drones. En el FDTSP se permite que los drones sean lanzados desde cualquier ubicación correspondiente a un cliente antendido por el camión. Así, los drones pueden, o no, retornar en el mismo lugar de lanzamiento para ser recepcionados, a diferencia del TSP-MD, donde los drones no pueden despegar y retornar en el mismo lugar. El camión está equipado con un número determinado de drones y no tiene limitaciones de carga ni de trayecto. La entrega de los envíos se realiza mediante el camión o los drones. Se busca minimizar el tiempo del recorrido, considerando las limitantes que los drones solo pueden llevar un paquete a la vez, y pueden realizar viajes según la capacidad de la batería. Debido a esto último, el camión puede esperar según corresponda la recepción de los drones.

En este estudio es abordado el FDTSP, mediante un modelo de programación lineal entera mixta (MILP por sus siglas en inglés: *Mixed Integer Linear Problem*) para su resolución. Además, debido a que es un problema de la clase *NP-Hard*, se propone una metaheurística para la resolución de este. El enfoque propuesto se caracteriza por construir una solución en dos niveles. El primer nivel es caracterizado por la optimización de la ruta del camión y en el segundo nivel es realizada la optimización de las rutas de los drones.

## 1.2. Objetivo general

Implementar un modelo y un algoritmo para el FDTSP.

## 1.3. Objetivos específicos

- Revisión del estado del arte del FDTSP y problemas similares.
- Diseñar un modelo de programación lineal entera mixta para el FDTSP.
- Diseñar una metaheurística para resolver el FDTSP.
- Implementar el modelo y el algoritmo propuesto.
- Analizar y comparar los resultados con el estado de arte.

## 1.4. Estructura del documento

El documento sigue con el Capítulo 2, donde se presenta formalmente el problema a tratar y una revisión de la literatura. Luego, en el Capítulo 3, se presenta el MILP propuesto. A continuación, el Capítulo 4 describe la descripción y la estructura de la metaheurística propuesta. Después, en el Capítulo 5, se presentan los experimentos computacionales y los resultados asociados. El Capítulo 6 analiza y discute los resultados obtenidos para concluir el trabajo realizado.

# Capítulo 2

# Problema flexible del vendedor viajero con múltiples drones

En este capítulo se describe formalmente el FDTSP y se realiza una revisión de la literatura asociada al problema.

## 2.1. Descripción del problema

El FDTSP es una variante del TSP-MD y es descrito con un grafo completo dirigido $G = (V, A)$. El conjunto de vértices es $V = \{0, 1, ..., n-1, n\}$ donde 0 y $n$ representan un único depósito, $N = \{1, ..., n-1\}$ representa el conjunto de clientes a atender. Mientras, el conjunto de arcos $A = \{(0, j)|j \in N\} \cup \{(i, j)|i, j \in N : i \neq j\} \cup \{(i, n)|i \in N\}$ disponen del tiempo asociado que representa el viaje realizado por un vehículo desde un vértice a otro. Además, $N^0 = N \cup 0$ y $N^N = N \cup n$ representan el conjunto de clientes a atender junto al depósito y su equivalente, respectivamente. Adicionalmente, el camión inicia el recorrido en el depósito 0 y termina en el depósito $n$. Este vehículo terrestre puede transportar $m$ drones idénticos, los cuales atienden un cliente por viaje. Cada viaje de un dron se representa por $L = \{(i, j, k)|i \in N^0, j \in N, k \in N^N : i \neq j \wedge k \neq j\} \setminus \{(0, j, n)|j \in N\}$.

Por otro lado, los vehículos tienen un tiempo de recorrido que se calcula según la velocidad a la que viajan en una distancia euclidiana. Así, los tiempos de viaje para cada arco son representados por $t^c_{i,j}$ y $t^d_{i,j}$, $(i, j) \in A$, para el camión y los drones, respectivamente. Cada cliente tiene un tiempo de atención dependiendo si es atendido por el camión ($s^c$) o

a algún dron ($s^d$). Es importante señalar que un cliente puede ser atendido por un único vehículo y, en situaciones donde varios vehículos convergen en un vértice, se considera que el camión lleva a cabo la atención. Además, el lanzamiento de un dron tiene un tiempo de preparación asociado ($t^{se}$) y cada dron no puede realizar un viaje mayor al tiempo limitado por su batería ($\varepsilon$), considerando distancias recorridas, tiempo de atención y tiempo de espera. Se asume que antes de cada lanzamiento el dron tiene la batería completa. Cuando un dron es lanzado desde un vértice y retorna a este mismo vértice, el camión debe permanecer en la ubicación hasta que el dron regrese. Mientras que si un vehículo llega a un vértice antes que la llegada del otro vehículo, este debe permanecer en el lugar hasta que el otro vehículo llegue. En caso de que algún dron tenga que esperar, este permanece en vuelo hasta la llegada de camión. Finalmente, el objetivo es encontrar las rutas entre el camión y una cantidad de drones que minimice el tiempo de finalización del recorrido.

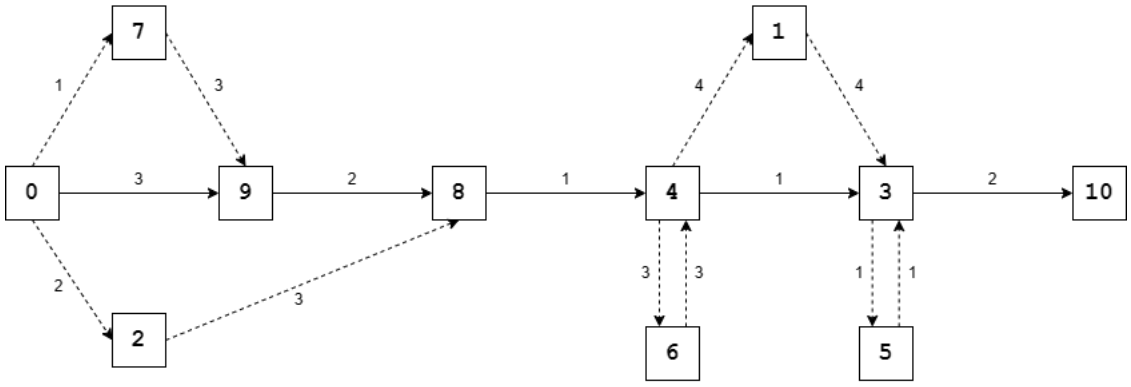


**Figura 1:** Representación de una solución factible para nueve clientes y dos drones.

En la Figura 1 se ilustra un ejemplo de una solución factible para el FDTSP con nueve clientes y dos drones. Los arcos representados con líneas continuas son parte de la ruta del camión y con líneas punteadas corresponden a las rutas de los drones. Los valores asociados a los arcos representan el tiempo que demoran los vehículos en viajar de un nodo a otro. Los drones tienen una batería de 60 minutos. Los dos drones son lanzados desde el depósito, y recepcionados en los clientes 9 y 8. Debido a los tiempos de viaje, el camión debe esperar la llegada del dron que vuelve desde el cliente 7, mientras que el dron es recepcionado en el cliente 8 debe esperar al camión. Luego, desde el cliente 4, los dos drones son lanzados nuevamente. Uno atiende al cliente 6 y retorna al mismo nodo de lanzamiento, mientras que el otro dron debe ser esperado por el camión en la ubicación del cliente 3. A continuación, un dron es lanzado hacia el cliente 5, y debe ser esperado

por el camión en la misma ubicación de lanzamiento. Finalmente, la ruta concluye con la llegada del camión y ambos drones al depósito. La Tabla 1 presenta los tiempos de llegada a los nodos para el ejemplo anteriormente descrito.

**Tabla 1:** Tiempos de llegada para cada nodo.

| **Vértice** | 0 | 7 | 2 | 9 | 8 | 4 | 6 | 1 | 3 | 5 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Tiempo de llegada [min]** | 0 | 3 | 4 | 6,5 | 8,5 | 10 | 15 | 16 | 20,5 | 22,5 | 26 |

## 2.2. Revisión de la literatura

Existen distintos enfoques para combinar la operación logística entre camiones y drones. Un sistema puede tener uno o varios camiones que realizan repartos en coordinación con un o múltiples drones. Además, existen sistemas en donde la operación de los drones es independiente del camión (Cavani *et al.*, 2021). Según Chung *et al.* (2020), la primera modalidad puede ser clasificada como operaciones combinadas entre camión y dron (DTCO por sus siglas en inglés: *Drone-Truck Combined Operations*), y la segunda modalidad es clasificada como operaciones de drones (DO por sus siglas en inglés: *Drone Operations*).

Los problemas presentados en esta revisión del estado del arte pertenecen a la clasificación DTCO, cambiando los nombres del problema según el autor del artículo revisado y algunos supuestos en la operación de los drones y el camión. Adicionalmente, se restringe la revisión a los problemas que toman supuestos no dinámicos. Estos supuestos pueden ser la ubicación en que se realizan los lanzamientos y recepciones de los drones, como en Salama & Srinivas (2022), donde las ubicaciones no se restringen solo a las posiciones de los clientes. Además, los supuestos dinámicos se pueden referir a que los vehículos atienden pedidos generados mientras están en ruta, como los ejemplos que se muestran en Zhang & Woensel (2023).

El primer estudio que combinó el TSP con drones fue propuesto por Murray & Chu (2015). Abordaron el FSTSP que consiste en un único camión de reparto que realiza una ruta de clientes asignados, mientras que un dron es lanzado desde un cliente visitado por el camión, en dirección a entregar un paquete a otro cliente. Luego, el dron se encuentra con el camión dentro de su ruta para ser recibido y preparado para el siguiente lanzamiento. Los autores proponen la implementación de un modelo de MILP y una heurística, demostrando con ambos enfoques la efectividad del uso de un camión y un dron para realizar entregas a una cantidad de 10 vértices.

Asimismo, Ha *et al.* (2018) toman el FSTSP propuesto por Murray & Chu (2015) y lo adaptan para minimizar el costo operacional del recorrido en vez del tiempo. Lo nombran TSP-D, pero por simplicidad en esta revisión se asocia con el FSTSP. Estos autores proponen un modelo de MILP, una heurística y un GRASP (por sus siglas en inglés: *Greedy Randomized Adaptive Search Procedure*) para resolver la adaptación. Los resultados computacionales muestran que el GRASP supera a los resultados de la heurística en instancias de tamaño entre 10 a 100 vértices. Mientras que el modelo puede encontrar soluciones óptimas de solo 10 vértices.

Luego, Agatz *et al.* (2018) definen el TSP-D. A diferencia de Ha *et al.* (2018), no existen tiempos asociados al lanzamiento del dron y se permite que el camión revisite algún cliente. De esta forma, para abordar el TSP-D proponen un modelo de programación lineal entero (ILP por sus siglas en inglés: *Integer Linear Programming*) y algoritmos heurísticos basados en búsquedas locales y programación dinámica. Los resultados muestran ahorros sustanciales con el enfoque TSP-D, en comparación al TSP. Además, se demuestra que el modelo resuelve de manera óptima instancias hasta con 12 vértices, mientras que las heurísticas son evaluados con instancias de tamaños que van desde 10 hasta 100 vértices.

En otro sentido, Roberti & Ruthmair (2021) proponen un modelo de MILP compacto y una resolución mediante programación dinámica combinada con un enfoque exacto basado en el *Branch and Price* (B&P) para abordar el TSP-D, y otras variantes de forma general. Diferenciándose de otros problemas en que no considera limitaciones de viaje para los drones y los clientes no tienen tiempos asociados a su atención. Abordan instancias de tamaño de 10 a 40 vértices. Sin embargo, el modelo obtiene resultados peores que los del estado del arte evaluado, pero verificando la sencillez de implementación. Mientras que en el enfoque de programación dinámica obtiene resultados superiores al estado del arte evaluado.

En base al estudio de Agatz *et al.* (2018), AlMuhaideb *et al.* (2021) proponen una metaheurística GRASP, que utiliza dos búsquedas locales y una selección adaptativa del vecindario a evaluar. Obteniendo resultados mejor o similares que los obtenidos por el estado de arte evaluado por ellos, considerando instancias de 10 a 100 vértices.

En la investigación realizada por Tu *et al.* (2018) extienden el FSTSP propuesto por Ha *et al.* (2018) a un TSP-MD, que considera la posibilidad de utilizar múltiples drones y un único camión. Los investigadores adaptan el GRASP propuesto por los autores

de referencia, además de proponer un ALNS (por sus siglas en inglés: *Adaptive Large Neighborhood Search*). Demostrando que ambas propuestas encuentran soluciones eficientes para instancias con 50 y 100 vértices, destacando que el ALNS obtiene mejores soluciones.

Más tarde, Cavani *et al.* (2021) a partir del TSP-D, diseñan una variante con múltiples drones (TSP-MD). Ellos proponen un modelo de MILP compacto, y lo abordan mediante un enfoque de descomposición exacta con un algoritmo de *Branch and Cut* (B&C). Mostrando que el modelo es útil para resolver de manera óptima instancias de hasta 25 vértices en menos de dos horas de cómputo. Superando a otros métodos exactos señalados en la literatura que solo pueden resolver instancias de hasta 10 vértices.

Recientemente, Tiniç *et al.* (2023) abordan el TSP-MD a partir del TSP-D, buscando minimizar el costo operacional de los recorridos, asociado al recorrido hecho por un único camión y múltiples drones. Este estudio se diferencia de los demás, en que los drones tiene permitido retornar a la misma ubicación de su lanzamiento para ser recepcionados por el camión. Adicionalmente, se compara la función objetivo de minimizar el costo operacional y el tiempo de llegada. Para resolver el problema proponen un modelo de MILP y un B&C. Los resultados son mejores en comparación con otras formulaciones exactas para instancias con 20 vértices, demostrando que minimizar el tiempo de llegada genera mejores soluciones que buscar minimizar el costo operacional.

De forma parecida al enfoque anteriormente descrito, Lu *et al.* (2022) basados en el TSP-D diseñan otro enfoque con múltiples drones que pueden realizar vuelos desde y hacia el mismo vértice de lanzamiento, con la diferencia que se busca minimizar el tiempo de recorrido. Para obtener soluciones, proponen un modelo de MILP que no es evaluado. Además, desarrollan un algoritmo de clusterización con el fin de asignar una cierta cantidad de clientes para ser atendidos por drones y por el camión, para luego optimizar la ruta del camión mediante una heurística. A esta metodología la llaman el FDTSP. Para optimizar la ruta del único camión proponen dos metahuerísticas para encontrar el mínimo tiempo de recorrido del camión. Las heurísticas son un algoritmo genético (GA por sus siglas en inglés: *Genetic Algorithm*) y un Simulated Annealing (SA). Además, utilizan el solver de OR-tools de Google. A pesar de que la evaluación realizada es solamente con los algoritmos y el solver, los resultados muestran reducciones significativas en los tiempos de operación comparando la realización de una ruta hecha solo por un camión, un TSP. Las instancias que evalúan tienen desde 15 a 150 vértices.

En cuanto a la utilización de múltiples camiones, Murray & Raj (2020) extienden el FSTSP, agregando la posibilidad de utilizar múltiples camiones. Abreviando el problema a mFSTSP (por sus siglas en inglés: *multiple Flying Sidekick Traveling Salesman Problem*), donde el objetivo es completar el recorrido para realizar entregas de paquetes en un mínimo tiempo. Proponen un modelo de MILP y una heurística de tres fases, el modelo obtiene resultados óptimos para la mayoría de las instancias con distintas capacidades de los drones y una cantidad de 9 vértices. Mientras que la heurística puede resolver de 9 hasta 101 vértices de manera efectiva, obteniendo ahorros importantes de tiempo de recorrido.

Posteriormente, Luo *et al.* (2021) implementan otro modelo de MILP, y una búsqueda tabú (TS por sus siglas en inglés: *Tabu Search*). Evalúan el desempeño del MILP con instancias de 8 y 10 vértices, pero estableciendo un límite de dos horas de cálculo. Mientras que el TS es evaluado con instancias desde 8 a 100 vértices. Los resultados muestran precisión y eficiencia del algoritmo propuesto en la resolución del problema, especialmente en instancias de tamaño pequeño, consiguiendo reducciones significativas en los tiempos de despacho para instancias con 8 vértices.

En esta misma línea, pero con la inclusión del uso de múltiples drones es que Kitjacharoenchai *et al.* (2019) extienden el TSP-MD y agregan la capacidad de utilizar múltiples camiones. Llamando a este enfoque mTSPD (por sus siglas en inglés: *Multiple Traveling Salesman with Drones*) y proponen un modelo de MILP y una heurística, realizando una comparación en instancias de 9 a 50 vértices. Además, comparan las soluciones entregadas por el algoritmo propuesto con la utilización de múltiples camiones sin la operación de drones utilizando instancias basadas en la biblioteca TSPLIB con hasta 100 vértices. Obteniendo resultados con mejoras sustanciales en las operaciones de despacho.

Finalmente, Kuo *et al.* (2023) analizan un caso más práctico que en otras investigaciones, enfocándose en compañías distribuidoras que buscan minimizar el tiempo de recorrido y las emisiones de contaminantes por el uso de los vehículos. Estas tienen en posesión un número considerable de camiones, así la idea es equipar a cada vehículo con un dron. En este estudio, los investigadores proponen la utilización de un modelo de MILP y un algoritmo NSGA-II (por sus siglas en inglés: *the second generation of Nondominated Sorting Genetic Algorithm*), con un enfoque múltiobjetivo. Sin embargo, la evaluación de los resultados se realiza únicamente con el NSGA-II con instancias que consideran desde 10 hasta 100 vértices. Obteniendo resultados que muestran la efectividad del algoritmo para los diversos

tamaños de instancias y reduciendo ambos objetivos en la rutas de los camiones y drones.

La Tabla 2 presenta un resumen de la literatura mencionada. En la columna "Problema" se específica la abreviación del problema que se plantean en la investigación, mientras que las columnas "# Camiones" y "# Drones", detallan la cantidad de camiones y drones utilizados en el problema revisado, respectivamente. La columna "Artículo" contiene las referencias del documento revisado, "Objetivo" es el resultado de interés del estudio, buscando minimizar el tiempo o el costo objetivo. La columna "Método exacto" hace referencia al uso de un MILP/ILP u otro enfoque. La columna "Algoritmo" se refriere al enfoque aproximado que es utilizado para resolver el problema. Finalmente, las columnas "$|V|$" tienen la cantidad de vértices presentes en cada problema.

**Tabla 2:** Resumen de la revisión de la literatura.

| Camiones | Drones | Problema | Artículo | Objetivo | Método exacto | \|V\| | Algoritmo | \|V\| |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | FSTSP | Murray & Chu (2015) | Tiempo | MILP | 10 | Heurística | 10 |
| | | | Ha *et al.* (2018) | Costo Operacional | MILP | 10 | GRASP | [10, 100] |
| | | TSP-D | Agatz *et al.* (2018) | Costo Operacional | ILP | 12 | Heurística | [10, 100] |
| | | | Roberti & Ruthmair (2021) | Tiempo | MILP, B&P | [10, 40] | - | - |
| | | | AlMuhaideb *et al.* (2021) | Tiempo | - | - | GRASP | [10, 100] |
| | m | TSP-MD | Tu *et al.* (2018) | Costo Operacional | - | - | GRASP,ALNS | [50, 100] |
| | | | Cavani *et al.* (2021) | Tiempo | MILP, B&C | 25 | - | - |
| | | | Tiniç *et al.* (2023) | Costo Operacional | MILP, B&C | 20 | - | - |
| | | FDTSP | Lu *et al.* (2022) | Tiempo | MILP | - | Heurística | [15, 150] |
| m | 1 | mFSTSP | Murray & Raj (2020) | Tiempo | MILP | 9 | Heurística | [9, 101] |
| | | MTSP-MD | Luo *et al.* (2021) | Tiempo | MILP | [8, 10] | TS | [25, 100] |
| | m | MTSP | Kitjacharoenchai *et al.* (2019) | Tiempo | MILP | [9, 50] | Heurística | [17, 100] |
| | | VRPD | Kuo *et al.* (2023) | Tiempo y Emisiones | MILP | - | NSGA-II | [10, 100] |

# Capítulo 3

# Modelo de programación lineal entera mixta

En este capítulo se presenta el modelo de programación lineal entera mixta propuesto y se describen el conjunto de parámetros, variables y restricciones asociadas al modelo planteado.

## 3.1. Modelo de MILP propuesto

El FDTSP puede ser modelado mediante un modelo de MILP. Esta propuesta se basa en el modelo de Cavani *et al.* (2021), considerando los supuestos necesarios para solucionar el FDTSP.

El MILP usa la variable de decisión binaria $x_{ij}$, $(i, j) \in A$, que toma el valor 1 si el camión se desplaza desde el vértice $i$ al $j$. Además, la variable binaria $y_i$ toma el valor 1 si el camión sirve al cliente $i \in N$. Asimismo, la variable binaria $z_{ijk}$, $(i, j, k) \in L$, que toma el valor 1 si el dron es lanzado desde el vértice $i$, sirve al cliente $j$ y es recepcionado en el vértice $k$. También, se define una variable entera, $w_i$ que representa el número de drones que están volando en el instante que el camión deja el vértice $i \in N$. Finalmente, la variable continua $a_i$ es el tiempo de llegada de un vehículo al vértice $i \in N$. La Tabla 3 presenta el resumen de los conjuntos, parámetros y variables.

**Tabla 3:** Resumen de los conjuntos, parámetros y variables de decisión.

| Conjuntos | Descripción |
|---|---|
| $V$ | Conjunto de vértices {0,1,2,...,$n-1$,$n$}. |
| $N$ | Conjunto de clientes {1,2,...,$n-1$}. |
| $N^0$ | Conjunto de clientes y el depósito {0,1,2,...,$n-1$}. |
| $N^N$ | Conjunto de clientes y la dúplica del depósito {1,2,...,$n-1$,$n$}. |
| $A$ | Conjunto de arcos. |
| $L$ | Conjunto de clientes visitados por un dron en un viaje. |
| **Parámetros** | **Descripción** |
| $M$ | Número muy grande. |
| $m$ | Número inicial de drones disponibles. |
| $\varepsilon$ | Tiempo en minutos de la duración de la batería de un dron. |
| $t^{se}$ | Tiempo de *setup* de un dron. |
| $s_i^c$ | Tiempo que dura el servicio del camión al cliente $i \in N$. |
| $s_i^d$ | Tiempo que dura el servicio de un dron al cliente $i \in N$. |
| $t_{ij}^c$ | Tiempo en que el camión tarda en desplazarse desde el vértice $i$ al $j$, $\forall (i,j) \in A$. |
| $t_{ij}^d$ | Tiempo en que el dron tarda en desplazarse desde el vértice $i$ al $j$, $\forall (i,j) \in A$. |
| **Variables de decisión** | **Descripción** |
| $x_{ij}$ | Representa el viaje del camión desde el vértice $i$ al $j$, $(i,j) \in A$. |
| $y_i$ | Representa que el cliente $i \in N$ es atendido por el camión. |
| $z_{ijk}$ | Representa el viaje de un dron, lanzado desde el cliente $i$, sirve al cliente $j$ y es recepcionado en $k$, $(i,j,k) \in L$. |
| $w_i$ | Número de drones volando en el instante que el camión deja el cliente $i \in N$. |
| $a_i$ | El tiempo en el que un vehículo llega al nodo $i \in N$. |

Así, el modelo de MILP para el FDTSP es presentado a continuación:

$$\min\ a_n \tag{1}$$

sujeto a:

$$\sum_{j \in N^0: j \neq i} x_{ji} = \sum_{j \in N^N: j \neq i} x_{ij} \qquad \forall i \in N \tag{2}$$

$$\sum_{j \in N^N: j \neq i} x_{ij} = y_i \qquad \forall i \in N \tag{3}$$

$$\sum_{j \in N} x_{0j} = \sum_{i \in N} x_{in} = 1 \tag{4}$$

$$y_j + \sum_{i \in N^0: i \neq j} \sum_{k \in N: k \neq j, k \neq i} z_{ijk} + \sum_{i \in N: i \neq j} z_{ijn} + \sum_{i \in N^0: i \neq j} z_{iji} = 1 \qquad \forall j \in N \tag{5}$$

$$y_i m \geq \sum_{j \in N: i \neq j} \sum_{k \in N^N: k \neq j} z_{ijk} \qquad \forall i \in N \tag{6}$$

$$m \geq \sum_{j \in N} \sum_{k \in N^0: k \neq j} z_{0jk} \tag{7}$$

$$y_i M \geq \sum_{j \in N: i \neq j} \sum_{k \in N^N: k \neq j} z_{ijk} + \sum_{k \in N^0} \sum_{j \in N: k \neq j, i \neq j} z_{kji} \qquad \forall i \in N \tag{8}$$

$$w_o = \sum_{j \in N} \sum_{k \in N^0: k \neq j} z_{0jk} - \sum_{j \in N} z_{0j0} \tag{9}$$

$$w_n = 0 \tag{10}$$

$$w_o \leq m \sum_{j \in N} x_{0j} \tag{11}$$

$$w_i \leq m \sum_{j \in N^N : j \neq i} x_{ij} \qquad \forall i \in N \tag{12}$$

$$w_0 - \sum_{r \in N : j \neq r} z_{0rj} + \sum_{r \in N : j \neq r} \sum_{s \in N^N : s \neq r, s \neq j} z_{jrs} \leq w_j + M(1 - x_{0j}) \qquad \forall j \in N \tag{13}$$

$$w_i - \sum_{s \in N^0 : s \neq r} \sum_{r \in N : j \neq r} z_{srj} + \sum_{r \in N : j \neq r} \sum_{s \in N^N : s \neq r, s \neq j} z_{jrs} \leq w_j + M(1 - x_{ij}) \qquad \begin{array}{l} \forall i \in N, \\ j \in N : i \neq j \end{array} \tag{14}$$

$$w_i - \sum_{s \in N} \sum_{r \in N : s \neq r} z_{srn} + \leq w_n + M(1 - x_{in}) \qquad \forall i \in N \tag{15}$$

$$\sum_{r \in N : j \neq r} \sum_{s \in N^N : s \neq r} z_{jrs} \leq m - w_i + \sum_{s \in N^0 : s \neq r, s \neq j} \sum_{r \in N : j \neq r} z_{srj} + M(1 - x_{ij}) \qquad \begin{array}{l} \forall i \in N^0, \\ j \in N : i \neq j \end{array} \tag{16}$$

$$a_0 = 0 \tag{17}$$

$$\begin{aligned} a_0 + (M + t^c_{0j}) x_{0j} + \sum_{k \in N : k \neq j} [M + t^d_{0j}] z_{0jk} + \sum_{r \in N} \sum_{s \in N : s \neq r} t^{se} z_{0rs} \\ + \max_{k \in N : i \neq k, j \neq k} [\max[t^d_{0k} + s^d_k + t^d_{kj} \ - t^c_{0j}, \ 0]] z_{0kj} \leq a_j + M \end{aligned} \qquad \forall j \in N \tag{18}$$

$$a_0 + (M + t^c_{0j}) x_{0j} + \max_{k \in N : i \neq k, j \neq k} [se + t^d_{0k} + s^d_k + t^d_{k0}] z_{0k0} \leq a_j + M \qquad \forall j \in N \tag{19}$$

$$\begin{aligned} a_i + (M + t^c_{ij} + s^c_i) x_{ij} + \sum_{k \in N^N : k \neq j} [M + s^c_i + t^d_{ij}] z_{ijk} \\ + \sum_{r \in N : i \neq r} \sum_{s \in N^N : s \neq r, s \neq i} t^{se} z_{irs} \\ + \max_{k \in N : i \neq k, j \neq k} [\max[t^d_{ik} + s^d_k + t^d_{kj} - t^c_{ij}, 0]] z_{ikj} \leq a_j + M \end{aligned} \qquad \begin{array}{l} \forall i \in N, \\ j \in N : \\ i \neq j \end{array} \tag{20}$$

$$\begin{aligned} a_i + (M + t^c_{ij} + s^c_i) x_{ij} \\ + \max_{k \in N : i \neq k, j \neq k} [t^{se} + t^d_{ik} + s^d_k + t^d_{ki}] z_{iki} \leq a_j + M \end{aligned} \qquad \begin{array}{l} \forall i \in N, \\ j \in N \end{array} \tag{21}$$

$$\begin{aligned} a_i + (M + t^c_{in} + s^c_i) x_{in} + \sum_{k \in N : k \neq i} (M + \max[t^c_{kn} - t^d_{ki}], s^d_i + t^d_{in}) z_{kin} \\ + \sum_{r \in N : i \neq r} \sum_{s \in N^N : s \neq r, s \neq i} t^{se} z_{irs} \\ + \max_{k \in N : i \neq k} [\max[t^d_{ik} + s^d_k + t^d_{kn} - t^c_{in}, 0]] z_{ikn} \leq a_n + M \end{aligned} \qquad \forall i \in N \tag{22}$$

$$0 \leq a_k - a_i + M(1 - z_{ijk}) \quad \forall i \in N^0, j \in N, k \in N^N : i \neq j, j \neq k, k \neq i \tag{23}$$

$$\varepsilon \geq t^d_{ij} + s^d_j + a_k - a_i - M(1 - z_{ijk}) \quad \forall i \in N^0, j \in N, k \in N : i \neq j \neq k \tag{24}$$

$$\varepsilon \geq t^d_{ij} + s^d_j + t^d_{ji} - M(1 - z_{iji}) \quad \forall i \in N^0, \; j \in N : \; i \neq j \tag{25}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{26}$$

$$y_i \in \{0, 1\} \quad \forall i \in N \tag{27}$$

$$z_{ijk} \in \{0, 1\} \quad \forall (i, j, k) \in L \tag{28}$$

$$w_i \in \mathbb{N}^0 \quad \forall i \in V \tag{29}$$

$$a_i \in \mathbb{R}_+ \quad \forall i \in V \tag{30}$$

La función objetivo (1) busca minimizar el tiempo de llegada del camión al depósito. Luego, las restricciones (2)–(3) buscan que el flujo de entrada y salida del camión por un cliente sea el mismo y que el cliente visitado por el camión sea atendido por este vehículo. Las restricciones (4) establecen que el camión debe salir y regresar al depósito. La restricciones (5) limitan que un clientes sea servido solamente por un vehículo. Las restricciones (6) y (7) limitan el número de lanzamientos desde un vértice o el depósito al numero máximo de drones disponibles. Las restricciones (8) limitan que un dron solamente puede ser lanzado o recepcionado en un vértice visitado por el camión. Las restricciones (9)-(10) limitan el número de drones volando desde el depósito. Mientras que las restricciones (11) y (12) limitan los vuelos desde y hacia los vértices según la presencia del camión. Las restricciones (13)-(16) balancean el número de drones volando según los lanzamientos y recepciones en cada vértice. La restricción (17) establece que el tiempo de inicio en el depósito es cero y así, las restricciones (18) y (19) pueden establecer los tiempos de viajes del camión y drones comenzando desde el depósito. Las restricciones (20) y (21) establecen los tiempos al igual que las restricciones anteriores, pero considerando que los traslados se realizan entre vértices correspondientes a clientes. Las restricciones (22) limitan los tiempos de llegada para traslados desde clientes al depósito final. Las restricciones (23)-(25) limitan los vuelos de los drones según el tiempo del recorrido y la duración de sus batería. Finalmente, las restricciones (26)-(30) definen los dominios de las variables de decisión.

Adicionalmente, $M$ corresponde a un número muy grande. Este es calculado como la suma de todas las distancias del camión y la suma de todas las distancias de los drones.

La Ecuación (31) representa el cálculo de $M$.

$$\sum_{(i,j)\in A} t_{ij}^{c} + \sum_{(i,j)\in A} t_{ij}^{d} \tag{31}$$

Respecto a la complejidad computacional del modelo de MILP propuesto, se utilizan $|V|$ variables continuas ($a$), $|V|$ variables enteras ($w$), $|V|^3 + |V|^2 + |V|$ variables binarias ($z$, $x$ y $y$, respectivamente). Por otro lado, el conjunto las restricciones tienen distintas complejidades computacionales como muestra el resumen de la Tabla 4. De manera general, las restricciones correspondientes al ruteo del camión presentan en el peor caso complejidad cúbica. Mientras que las restricciones que balancean el número de drones en vuelo y las restricciones de tiempo de llegada tienen en el peor caso una complejidad cuártica. Finalmente, la complejidad final de las restricciones es $\mathcal{O}(|N^4|)$.

**Tabla 4:** Resumen de la complejidad computacional de las variables y restricciones.

| **Complejidad computacional** | **Restricciones** |
|---|---|
| $\mathcal{O}(1)$ | (1),(10),(17) |
| $\mathcal{O}(\|N\|)$ | (4),(11) |
| $\mathcal{O}(\|N\|^2)$ | (2),(3),(7),(9),(12),(19),(25) |
| $\mathcal{O}(\|N\|^3)$ | (5),(6),(8),(13),(15),(18),(21),(22),(23) |
| $\mathcal{O}(\|N\|^4)$ | (14),(16),(20),(24) |
| **Complejidad Final** | $\mathcal{O}(\|N\|^4)$ |

# Capítulo 4

# Metaheurística

Este capítulo presenta el algoritmo propuesto para resolver el problema, basado en un VNS. Adicionalmente, se describen los operadores utilizados de búsquedas en vecindario y las perturbaciones.

## 4.1. Estructura general

Para abordar el FDTSP de manera aproximada en este estudio, se propone una adaptación del algoritmo VNS introducido por Mladenović & Hansen (1997). Estos algoritmos exploran distintos vecindarios de un solución y analizan distintos vecindarios en busca de mejoras. En cada iteración son aplicadas búsquedas locales y perturbaciones para explorar distintos vecindarios y escapar de los óptimos locales.

El VNS propuesto opera a partir de la construcción de una solución inicial. Esta solución es optimizada en dos fases con el fin de encontrar la mejor combinación entre la ruta del camión y múltiples viajes de drones. El algoritmo propuesto es presentado en el pseudocódigo del Algoritmo 1.

La solución inicial es generada mediante un procedimiento heurístico que garantiza la construcción de una solución factible (línea 1). Luego, en las líneas 4-29, se entra al ciclo principal del algoritmo propuesto hasta que se cumpla el número de iteraciones máximas ($I_1$). Dentro del ciclo principal, la primera y segunda fase de optimización son llevadas a cabo según los parámetros $c_1$ y $c_2$, respectivamente. La primera fase consiste en reasignar la cantidad de clientes a ser atendidos por el camión o los drones (líneas 5-17). Una vez

hecha la reasignación, se acepta la solución con una probabilidad asociada al parámetro $\rho_2$, sin importar que exista o no alguna mejora en la solución (líneas 7-9). A partir de esto, se optimiza la ruta del camión aplicando una perturbación y una búsqueda local, estas son aplicadas una cantidad de veces determinada por el parámetro $K_1$. Mientras, en las líneas 18-27, la segunda fase utiliza un SA como heurística de búsqueda para las rutas de los drones, siempre que existan clientes asignados a ser visitados por algún dron. Luego, si se encuentra una solución que reduzca el tiempo de recorrido para atender a todos los clientes, entonces, se procede a actualizar la mejor solución conocida (líneas 24-26).

En el algoritmo se pueden distinguir dos tipos de cálculo del costo de la solución, equivalente al tiempo de recorrido de las rutas hasta el depósito. En la primera fase se determina solamente el tiempo que demora el camión en visitar a los clientes designados para este vehículo, equivalente a calcular el costo de una ruta TSP (línea 13). Por otro lado, en la segunda fase de optimización el costo de la solución es obtenido evaluando el tiempo de recorrido del camión en conjunto con los drones usados para atender a los clientes, distinguiendo si una solución es factible o no (línea 24). A diferencia de las otras dos fases, en la construcción de la solución inicial, el cálculo del costo es realizada de ambas formas según si la solución generada es factible o no.

En las siguientes secciones es presentada la representación de una solución, el procedimiento de penalización de infactibilidad de una solución. Además, se detalla cada uno de los operadores utilizados en las fases de la construcción inicial, la optimización de la ruta del camión y las rutas de los drones.

**Algoritmo 1:** Variable Neighborhood Search (VNS)

```
1  s̄ = solución-inicial()
2  s* = s̄
3  i ⟵ 1
4  while i ≤ I_1 do
5      if i%c_1 = 0 then                                   /* Primera fase */
6          s = reasiganción(s̄, ρ_1)
7          if random(0, 1) ≤ ρ_2 then
8              s̄ = s
9          end
10         for k = 1 to K_1 do
11             s = cambio-orden-camión(s)
12             s = 2-opt(s)
13             if f_tsp(s) ≤ f_tsp(s̄) then
14                 s̄ = s
15             end
16         end
17     end
18     if i%c_2 = 0 then                                   /* Segunda fase */
19         if hay clientes asignados para los drones then
20             s = SA(s̄)
21         else
22             s = s̄
23         end
24         if f(s) ≤ f(s*) then
25             s* = s̄ = s
26         end
27     end
28     i ⟵ i + 1
29 end
```

## 4.2. Representación de una solución

Una solución del problema está representada por $s$ que consiste en una tupla, $s = \{s^C, s^D\}$. Donde, $s^C$ representa un ciclo hamiltoniano realizado por el camión y $s^D$ son las rutas realizadas por los drones. $s^C$ es una lista que contiene los vértices visitados exclusivamente para el camión en orden cronológico, siendo el depósito el primer y el último elemento en la lista. Mientras, $s^D$ es una lista que almacena las rutas de los drones siguiendo una tripleta: $\{i, j, k\}$, donde $i$ y $k$ representan el vértice de lanzamiento y de recepción, que necesariamente tienen que ser visitados por el camión y $j$ es el vértice de un cliente que es atendido por el dron.

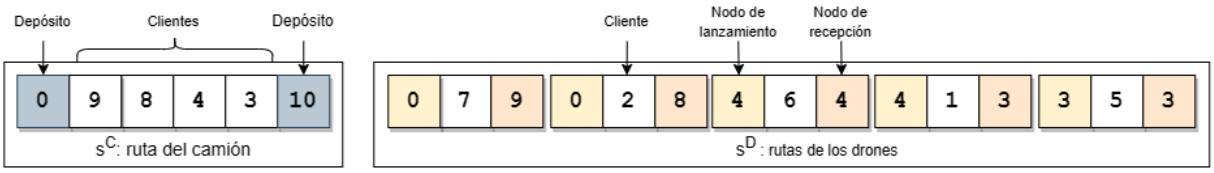


**Figura 2:** Representación de una solución para nueve clientes y dos drones.

La Figura 2 ilustra la representación para una solución factible. Esta solución representada es para el problema con nueve clientes y dos drones mostrado en la Figura 1. El elemento $s^C$ de la solución contiene los vértices visitados secuencialmente por el camión. Partiendo por el depósito, visitando 4 clientes y retornando al duplicado del depósito. Mientras que el elemento $s^D$ contiene las tripletas que representan los viajes de los drones. En cada tripleta se pueden distinguir los nodos de lanzamiento y recepción, que también pertenecen a la lista $s^C$, y el cliente atendido por el dron en el viaje.

## 4.3. Procedimiento de penalización de soluciones infactibles

Como se señaló anteriormente, existe la posibilidad de que una solución generada sea infactible, específicamente la infactibilidad puede ocurrir dentro de alguna tripleta construida para la ruta de los drones. Esto debido a que las secuencias utilizadas están diseñadas para operar principalmente en base a la aleatoriedad. Por tanto, si una solución contiene una tripleta infactible, esta solución es considerada infactible.

Se pueden dar cuatro tipos de infactibilidades en una tripleta: i) si una ruta de un dron

comienza en un nodo y retorna a un nodo que el camión ya visitó. ii) si el número de drones lanzados es mayor a la cantidad disponible en ese instante. iii) si el número de drones recepcionados excede a la cantidad lanzada con anterioridad. Finalmente, iv) si la duración del viaje de un dron es mayor a lo permitido por la batería.

Una solución infactible es identificada al calcular el costo del recorrido de la operación entre el camión y los drones. Se aborda de manera que sea comparable con las soluciones factibles encontradas. Debido que el elemento $s^D$ de una solución puede contener múltiples tripletas, existe la posibilidad de que exista una infactibilidad por cada tripleta. De esta manera, al calcular el costo del recorrido se emplea una penalización con el fin de comparar soluciones, subestimando las infactibles. Por cada infactibilidad encontrada dentro de una solución se suma una constante al tiempo de llegada del camión al depósito. En la Ecuación (32) se especifica el cálculo del costo de una solución infactible ($s$), donde $q$ es la cantidad de infactibilidades presentes en la solución y $P$ es la constante de penalización.

$$f(s) = \texttt{tiempo-llegada-depósito} + qP \tag{32}$$

## 4.4. Procedimiento de solución inicial

Para generar una solución inicial es propuesto un enfoque de cuatro fases. La primera consiste en la asignación inicial de los clientes, luego se sigue con la construcción inicial de la ruta del camión. La tercera fase es la construcción de las rutas para los drones. Finalmente, en caso que una solución sea infactible se procede con la cuarta fase, que consiste en construir una ruta del camión que atienda a todos los clientes. El procedimiento de solución inicial se ilustra en el Algoritmo 2.

**Algoritmo 2:** solución-inicial

```
1 s^C, s^D = asignación-aleatoria()
2 s^C = LKH(s^C)
3 s^D = construcción-ruta-drones(s^C, s^D)
4 s = {s^C, s^D}
5 if s es infactible then
6 |   s = LKH()
7 end
```

El procedimiento comienza asignando aleatoriamente los vértices visitados por el camión

y por los drones en la línea 1. La cantidad de clientes asignados para ser atendidos por los drones corresponde a la cantidad de drones inicialmente disponibles. Posteriormente, teniendo los clientes asignados, se procede a construir la ruta para el camión (línea 2). Esto se realiza resolviendo un TSP para todos los vértices del camión, utilizando la heurística Lin–Kernighan-Helsgaun (LKH) (Helsgaun, 2000), basada en la heurística clásica de Lin & Kernighan (1973). LKH es caracterizada por encontrar una buena aproximación a la ruta óptima con un costo computacional efectivo. Luego, conociendo la secuencia de vértices visitados por el camión se construyen las rutas de los drones (línea 3). Inicialmente, en base a los clientes asignados a ser atendido por los drones, se asignan aleatoriamente como nodos de lanzamiento y/o de recepción que conforman la tripleta. Seguidamente, se aplica una búsqueda local consistiendo en evaluar el intercambio de los nodos de lanzamiento o recepción, confirmando el cambio al encontrar una mejora en la solución.

La búsqueda local `construcción-ruta-drones()` consiste en evaluar cada ruta o tripleta generada y verificar si con intercambios sucesivos de los vértices de lanzamiento y recepción existe una reducción del tiempo del recorrido de los drones. Con el fin de reducir la posibilidad de obtener soluciones infactibles, el intercambio de los vértices de lanzamiento y de recepción se realiza individualmente. La evaluación de los vértices de lanzamiento consiste en examinar aquellos puntos que pertenecen a la ruta del camión y son visitados con anterioridad al vértice de recepción. Mientras que la evaluación de los puntos de recepción, es realizada con los vértices que son visitados por el camión después del nodo de lanzamiento. La evaluación termina una vez que se haya encontrado la primera reducción del tiempo de recorrido.

Para terminar con la construcción inicial, se evalúa si la solución encontrada es infactible. En el caso de ser infactible, entonces, se emplea nuevamente la heurística de LKH para encontrar una única ruta del camión que cubra a todos los clientes. Mientras que si la solución encontrada es factible, esta es aceptada y pasa a la fase siguiente del algoritmo principal. De esta forma, este procedimiento asegura que la solución inicial siempre sea factible.

## 4.5. Optimización de la ruta del camión

Esta primera fase del Algoritmo 1 es realizada para mejorar la ruta del camión actual. Consiste en la aplicación de dos perturbaciones y una búsqueda local. Las dos perturbaciones consisten en una reasignación aleatoria de los clientes atendidos por los

vehículos, `reasignación()`. En tanto, la segunda es un cambio en el orden de los clientes visitados por el camión, `cambio-orden-camión()`. Mientras, la búsqueda local empleada es la heurística clásica 2-opt. A continuación, se describe detalladamente los procedimientos anteriormente mencionados.

- `reasignación()`: este procedimiento reasigna los clientes atendidos por lo vehículos, con el objetivo perturbar los vecindarios de búsqueda tanto del recorrido del camión como de los drones. Para ello, se incrementa la lista de clientes visitados por los drones o por el camión. Según una probabilidad asociada al parámetro $\rho_1$, la ruta de los drones es ampliada con la selección aleatoria de un cliente atendido por el camión, para luego, sea asignado a ser atendido por algún dron. Mientras que la ruta del camión es ampliada con la selección aleatoria de un nodo atendido por algún dron y posteriormente, este es insertado aleatoriamente dentro de la secuencia de la ruta del camión.

- `cambio-orden-camión()`: Esta tiene el propósito de cambiar la secuencia de los clientes atendidos por el camión. Este proceso se realiza de manera aleatoria, intercambiando la posición de dos vértices seleccionados dentro de la ruta del camión, excluyendo a los vértices correspondientes al depósito.

- `2-opt()`: Se utiliza el algoritmo 2-opt propuesto por Croes (1958) como búsqueda local, con el propósito de mejorar el recorrido realizado por el camión. Esta clásica heurística consiste en intercambiar sucesivamente dos arcos que se intersectan. La idea es mejorar la ruta del camión reduciendo el tiempo de recorrido, de forma que este se acerque al costo óptimo.

## 4.6. Optimización de las rutas de los drones

Esta fase consiste en adaptar la metaheurística SA introducida por Kirkpatrick *et al.* (1983). Este algoritmo está basado en la analogía a los procesos de metalurgia. Donde a partir de un estado inicial se realiza una búsqueda de una mejor solución en base a una perturbación, y se decide permanecer en el vecindario en caso de encontrar una mejora o cambiar a otro vecindario en base a una probabilidad. De esta manera, se propone el pseudocódigo que se presenta en el Algoritmo 3, que consiste en generar una solución inicial y a partir de esta se aplica una perturbación en busca de mejores resultados.

**Algoritmo 3:** SA

```
1  d = construcción-ruta-drones(s^C, s^D)
2  d* = d
3  i ⟵ 1
4  t ⟵ t_0
5  while T ≤ t and i ≤ I_2 do
6      d̄ = d
7      for k = 1 to K_2 do
8          d = cambio-tripleta(d, σ_1, σ_2)
9          if f(d) < f(d̄) then
10             d̄ = d
11             if f(d̄) < f(d*) then
12                 d* = d̄
13             end
14         else if random(0,1) < e^{-|f(d̄)-f(d)|/t} then
15             d̄ = d
16         end
17     end
18     i ⟵ i + 1
19     t ⟵ αt
20 end
```

Para la optimización de la ruta de los drones se requiere conocer la ruta del camión y los clientes asignados a ser atendidos por algún dron, obtenidos en las fases anteriores. A partir de esto, y sin modificar la ruta del camión, se construye la lista inicial que contenga las tripletas de para cada viaje de un dron ($d$), esta construcción es la misma utilizada en la fase de la construcción de la solución inicial presentada en la línea 3 del Algoritmo 2. Esta consiste en asignar aleatoriamente un nodo de lanzamiento y de recepción para que un dron visite a un cliente, luego se aplica una búsqueda local que termina al encontrar la primera mejora. La búsqueda sigue hasta que se cumplen los criterios de la línea 5. Dentro de cada iteración, línea 7-17, se aplica una perturbación (línea 8) y se evalúa si existe una mejora, realizando un cambio de vecindario de búsqueda de manera aleatoria (líneas 9-16). El proceso de perturbar y visitar distintos vecindarios es realizada en base al parámetro $K_2$.

El proceso de perturbación `cambio-tripleta()` consiste en intercambiar dentro de cada tripleta aleatoriamente el vértice de lanzamiento, el vértice de recepción, o ambos por un

vértice que es visitado por el camión (línea 8). El cambio de ambos vértices tiene una probabilidad relacionada con el parámetro $\sigma_1$. La elección del intercambio del nodo de lanzamiento o de recepción se realiza en base a una probabilidad del parámetro $\sigma_2$. Estos procedimientos tienen asociadas probabilidades con el propósito de reducir la posibilidad de generar una infactibilidad y que el cambio de vecindario realizado no sea abrupto para la solución.

Mientras que la evaluación de la solución consiste en determinar si el costo de la ruta generada por la perturbación, es mejor que el costo actual conocido. En caso de ser mejor el costo, entonces, la mejor solución conocida es actualizada (líneas 9-13). En caso de que no exista una mejora con respecto a la solución actual, existe la posibilidad de aceptar la solución determinada por la evaluación de un cociente entre la diferencia de los costos y el parámetro $t$ en una función exponencial (líneas 14-15).

# Capítulo 5

# Experimentos computacionales

En este capítulo se presentan las instancias utilizadas para evaluar la efectividad del modelo de MILP y el VNS propuestos. Además, de realizar una comparación con los algoritmos de la literatura.

## 5.1. Descripción de las instancias

El conjunto de las instancias (*Set large*) usadas son propuestas por Lu *et al.* (2022), se basan en condiciones de zonas urbanas, suburbanas y rurales. En cada zona los clientes están distribuidos aleatoriamente. Las zonas urbanas consisten en un área de 10×10 unidades, las cuales pueden tener 80, 100 o 150 vértices. Las zonas suburbanas corresponden a un área 20×20 unidades, que pueden albergar 40, 60 o 80 vértices. Finalmente, las zonas rurales de un área de 40×40 unidades, pueden tener 15, 30 o 40 vértices. La Tabla 5 resume este set de instancias, presentando los distintos escenarios, sus áreas y el número de vértices ($|V|$).

**Tabla 5:** Resumen del *Set large* de instancias propuesto por Lu *et al.* (2022).

| Escenario | Área [$milla^2$] | $\|V\|$ |
|---|---|---|
| Urbano | 10×10 | {80, 100, 150} |
| Suburbano | 20×20 | {40, 60, 80} |
| Rural | 40×40 | {15, 30, 40} |

Además, con el fin de comparar el modelo de MILP propuesto se propone un nuevo conjunto de instancias (*Set small*). Este set consiste en las mismas tres condiciones del

conjunto de instancias anteriormente descrito, pero se reduce la cantidad de vértices presentes. Cada zona puede tener 5, 8, 12, 15, 18 o 20 nodos. La Tabla 6 resume este conjunto de instancias.

**Tabla 6:** Resumen del *Set small* de instancias.

| Escenario | Área [$milla^2$] | $\|V\|$ |
|---|---|---|
| Urbano | 10×10 | |
| Suburbano | 20×20 | {5, 8, 12, 15, 18, 20} |
| Rural | 40×40 | |

Cada instancia es evaluada con el modelo y algoritmo propuestos considerando 1, 2, 3, 4 y 5 drones, los cuales pueden tener una velocidad de 40, 60 y 80 millas por hora y una duración de la batería de 30 o 60 minutos. En la Tabla 7 se resumen los parámetros descritos.

**Tabla 7:** Resumen de los parámetros para los conjuntos de instancias.

| Numero de drones | Duración de la batería [min] | Velocidad de un dron [mph] | Velocidad del camión [mph] |
|---|---|---|---|
| {1, 2, 3, 4, 5} | {30, 60} | {40, 60, 80} | {35} |
| Tiempo de atención por cliente [min] | | Tiempo de preparación para un lanzamiento [min] | |
| 0,5 | | 1 | |

Cabe destacar que son 270 instancias para el *Set large* y 540 instancias para el *Set small*, resultando en un total de 810 instancias.

## 5.2. Configuración de los experimentos computacionales

El algoritmo y el modelo de MILP fueron implementados en Python 3.8.7. Los experimentos fueron ejecutados en el NLHPC, utilizando 2 procesadores Intel Xean Gold 6152 con 2.10 GHz, cada uno con 22 núcleos y 769 GB de RAM. Todos los experimentos fueron conducidos en el sistema operativo CentOS Linux 7 (64-bit). Para resolver el modelo MILP propuesto es utilizado el solver IBM ILOG CPLEX 12.9.

## 5.3. Calibración de los parámetros

El VNS propuesto utiliza un total de 14 parámetros. Dada la cantidad de parámetros, el desempeño del algoritmo puede verse afectado, si no se ajusta de manera adecuada los

parámetros. Por esto, se realiza una calibración de los parámetros con el paquete diseñado por López-Ibáñez *et al.* (2016) (IRACE). Se utilizan 20 instancias seleccionadas del *Set large* en base a experimentación previa, escogiendo instancias de cada escenario que presentaron peor rendimiento. La ejecución de IRACE consideró un nivel del 95 % de confianza, 10000 iteraciones y tomó un tiempo de 49,7 horas aproximadamente. La calibración fue realizada en el NLHPC usando 40 hilos en paralelo. Los parámetros calibrados, su descripción, rango y valores obtenidos por IRACE están listados en la Tabla 8.

**Tabla 8:** Rango y valores de los parámetros.

| Parámetro | Descripción | Rango | Valor |
|---|---|---|---|
| $I_1$ | Número de iteraciones del VNS. | {100000} | 100000 |
| $I_2$ | Número de iteraciones del SA. | {10, 20, 30, 40, 50, 60, 70, 80, 90, 100} | 80 |
| $K_1$ | Tamaño del vecindario del VNS. | [1, 5] | 4 |
| $K_2$ | Tamaño del vecindario del SA. | [1, 5] | 5 |
| $c_1$ | Constante para el módulo de la primera fase. | {50, 100, 150, 200, 250, 300, 350, 400, 450, 500} | 100 |
| $c_2$ | Constante para el módulo de la segunda fase. | {50, 100, 150, 200, 250, 300, 350, 400, 450, 500} | 100 |
| $\rho_1$ | Probabilidad asociada al operador `reasignación()`. | [0, 1] | 0,992 |
| $\rho_2$ | Probabilidad asociada para aceptar la reasignación. | [0, 1] | 0,842 |
| $\alpha$ | Constante de descenso de la temperatura del SA. | [0,95, 1] | 0,999 |
| $t_0$ | Temperatura inicial del SA. | {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150} | 30 |
| $T$ | Temperatura final del SA. | $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}\}$ | $10^{-8}$ |
| $\sigma_1$ | Probabilidad asociada al operador `cambio-tripleta()` para el intercambio del vértice de lanzamiento y de recepción. | [0, 1] | 0,326 |
| $\sigma_2$ | Probabilidad asociada al operador `cambio-tripleta()` para el intercambio único del vértice de lanzamiento o de recepción. | [0, 1] | 0,331 |
| $P$ | Constante de penalización. | {500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000} | 2000 |

## 5.4. Medida de calidad de la solución

De los resultados obtenidos por los algoritmos, se registran los valores mínimos, su promedio y el tiempo medio de ejecución. Para las instancias del *Set large*, la medida de calidad de las soluciones encontradas es la diferencia porcentual (gap) entre la solución encontrada por el VNS propuesto o alguno de los algoritmos de Lu *et al.* (2022) (SA o GA) y la mejor solución encontrada por alguno de estos algoritmos. Esto significa que, teniendo los resultados para cada instancia se selecciona el mejor costo como referencia para la comparación. La fórmula del gap se describe en la Ecuación (33), donde $f(s^*)$ corresponde al mejor costo encontrado por algunos de los tres algoritmos y $f(s)$ es el costo encontrado por cada algoritmo. Este gap es calculado para el costo mínimo ($\texttt{gap}_{\texttt{min}}$) y costo medio ($\texttt{gap}_{\texttt{avg}}$) obtenido para cada instancia por los algoritmos.

$$\frac{f(s) - f(s^*)}{f(s^*)} \times 100\,\% \tag{33}$$

Además, se mide la diferencias entre los tiempos de ejecución de los tres algoritmos y el

límite de 10 segundos establecidos. De esta manera, $\texttt{gap}_{\texttt{t}}$ es calculado entre el límite de 10 segundos y el tiempo medio requerido por las heurísticas para cada instancia ($t$). La Ecuación (34) representa el gap descrito.

$$\frac{t-10}{10} \times 100\,\% \tag{34}$$

Finalmente, se utilizan las métricas $\texttt{hit}^{\texttt{large}}_{\texttt{avg}}$ y $\texttt{hit}^{\texttt{large}}_{\texttt{min}}$, que consisten en un conteo de la cantidad de veces que un algoritmo alcanza un gap igual a cero para cada instancia, para el gap de los resultados medio y mínimos, respectivamente.

Para las instancias del *Set small*, la medida de calidad de las soluciones encontradas por el modelo de MILP es el $\texttt{gap}_{\texttt{MILP}}$ entre el límite superior ($UB$ por sus siglas en inglés: Upper Bound) y el límite inferior ($LB$ por sus siglas en inglés: Lower Bound). Este gap se muestra en la Ecuación (35).

$$\frac{UB-LB}{LB} \times 100\,\% \tag{35}$$

Mientras que la medida de calidad entre las soluciones encontradas para este conjunto de instancias corresponde también a la Ecuación (33). La diferencia está en que se consideran tanto los costos mínimos de los algoritmos, como los límites superiores encontrados por el modelo. De esta manera $f(s^*)$ corresponde al mejor costo encontrado y $f(s)$ es el costo encontrado por cada método.

De igual manera, para este conjunto de instancias se usa la métrica $\texttt{hit}^{\texttt{small}}_{\texttt{min}}$. Correspondiente al conteo de la cantidad de veces que el modelo o un algoritmo alcanza un gap igual a cero para cada instancia.

## 5.5. Resultados de los experimentos computacionales

Con el fin de evaluar en un contexto justo, los algoritmos VNS, SA y GA son ejecutados 10 veces de forma independiente, con distintas semillas aleatorias, para cada instancia. Además, cada ejecución se limitó a un tiempo de 10 segundos para las instancias *Set large* y *Set small*. Esto último quiere decir que pasado los 10 segundos no se vuelve a realizar una iteración del algoritmo, pero se permite terminar la última iteración iniciada antes del

tiempo límite.

Por otro lado, para el MILP se establece un tiempo límite de una hora (3600 segundos) de ejecución. Si en este periodo la solución óptima no es encontrada, se utiliza el costo de la mejor solución factible encontrada.

Debido a que el *Set large* consta de 270 instancias y el *Set small* de 540 instancias, se resumen los resultados agrupándolos por el número de vértices según escenario. De esta forma en las tablas se presentan los valores de las métricas promediados según el escenario y las instancia que contienen la misma cantidad de clientes. Las tablas muestran los resultados considerando una cantidad determinada de drones, por esto en cada tabla están agrupadas 54 instancias para el *Set large* y 108 para el *Set small*.

### 5.5.1. Resultados del *Set large*

Las Tablas 9–13 presentan la comparación de los resultados del VNS, SA y GA para las instancias del *Set large* considerando los resultados para uno, dos, tres, cuatro y cinco drones, respectivamente. Todas las tablas mencionadas tienen el siguiente formato: la primera columna se refiere al tamaño del plano para los tipos de escenario Urbano (U), Suburbano (S) y Rural (R). La segunda columna indica la cantidad de vértices presentes en la instancia. Luego, la tercera, cuarta y quinta columna corresponden al gap promedio ($\texttt{gap}_{\texttt{avg}}$) de los algoritmos VNS, SA y GA, respectivamente. En las siguientes columnas se repite el mismo formato anterior para el gap mínimo ($\texttt{gap}_{\texttt{min}}$). Desde la octava columna se presentan los hits obtenidos por el VNS, SA, y GA para los resultados medio ($\texttt{hit}^{\texttt{large}}_{\texttt{avg}}$). Mientras que a partir de la doceava columna se muestra la cantidad de hits obtenidos para los resultados mínimos ($\texttt{hit}^{\texttt{large}}_{\texttt{min}}$), siguiendo el mismo orden que las columnas anteriores. Las últimas tres columnas presentan el el gap del tiempo promedio de ejecución ($\texttt{gap}_{\texttt{t}}$). Finalmente, la penúltima fila muestra el promedio de las columnas de cada algoritmo. Mientras que la última fila muestra la totalidad de hits obtenidos por un algoritmo, así el número máximo de hits de cada algoritmo es de 54, equivalente a la cantidad total de instancias considerando una cantidad de drones específica.

**Tabla 9:** Comparación de los algoritmos considerando un dron.

| Tipo | \|V\| | $\texttt{gap}_{\texttt{avg}}$ (%) | | | $\texttt{gap}_{\texttt{min}}$ (%) | | | $\texttt{hit}_{\texttt{avg}}^{\texttt{large}}$ | | | $\texttt{hit}_{\texttt{min}}^{\texttt{large}}$ | | | $\texttt{gap}_{\texttt{t}}$ (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** |
| **U** | **80** | 0,00 | 9,46 | 24,14 | 0,00 | 5,47 | 18,27 | 5 | 1 | 0 | 4 | 2 | 0 | 10,07 | 6,30 | 6767,98 |
| | **100** | 0,00 | 13,94 | 36,15 | 0,00 | 9,13 | 28,61 | 6 | 0 | 0 | 5 | 1 | 0 | 9,90 | 7,63 | 13062,60 |
| | **150** | 0,00 | 11,86 | 59,35 | 0,00 | 8,24 | 48,93 | 5 | 1 | 0 | 5 | 1 | 0 | 14,39 | 11,47 | 51799,96 |
| **S** | **40** | 0,00 | 5,55 | 10,12 | 0,00 | 3,50 | 3,94 | 4 | 2 | 0 | 4 | 1 | 1 | 7,51 | 3,22 | 1634,83 |
| | **60** | 0,00 | 9,39 | 17,43 | 0,00 | 5,99 | 11,34 | 5 | 1 | 0 | 4 | 2 | 0 | 9,32 | 4,68 | 3952,09 |
| | **80** | 0,00 | 14,44 | 30,46 | 0,00 | 10,20 | 20,56 | 5 | 1 | 0 | 4 | 2 | 0 | 11,58 | 6,19 | 8211,64 |
| **R** | **15** | 0,00 | 3,36 | 5,46 | 0,00 | 7,31 | 7,13 | 4 | 2 | 0 | 5 | 0 | 1 | 1,16 | 1,69 | 254,91 |
| | **30** | 0,00 | 3,52 | 9,80 | 0,00 | 2,44 | 5,60 | 4 | 2 | 0 | 5 | 1 | 0 | 3,41 | 2,93 | 784,02 |
| | **40** | 0,00 | 6,23 | 12,15 | 0,00 | 4,07 | 7,19 | 5 | 1 | 0 | 4 | 2 | 0 | 5,12 | 3,23 | 1458,36 |
| Promedio | | 0,00 | 8,64 | 22,78 | 0,00 | 6,26 | 16,84 | 4,78 | 1,22 | 0,00 | 4,44 | 1,33 | 0,22 | 8,05 | 5,26 | 9769,60 |
| **Suma de las 54 instancias** | | | | | | | | 43 | 11 | 0 | 40 | 12 | 2 | | | |

La Tabla 9 muestra el rendimiento de los algoritmos con un dron. Donde el VNS encuentra las mejores soluciones en $\texttt{gap}_{\texttt{avg}}$ y $\texttt{gap}_{\texttt{min}}$, lo que se refleja en los 43 y 40 hits obtenidos, respectivamente. El segundo que mejor soluciones obtiene es el SA, con un $\texttt{gap}_{\texttt{avg}}$ promedio de $8,64\,\%$ y un $\texttt{gap}_{\texttt{min}}$ promedio de $6,26\,\%$. Así, el GA es el algoritmo con peores resultados, con un promedio de $\texttt{gap}_{\texttt{avg}}$ y $\texttt{gap}_{\texttt{min}}$ de $22,78\,\%$ y $16,84\,\%$, respectivamente. Con respecto al tiempo límite de 10 segundos, el SA es la heurística que presenta una menor diferencia porcentual. Mientras que el GA presenta una desviación más grande con el tiempo límite, excediendo notoriamente los 10 segundos. El VNS tiene un promedio de ejecución que se desvía en promedio en un $8,05\,\%$. Ninguno de los algoritmos logra detenerse antes de los 10 segundo, indicando que al cumplir el tiempo límite estos algoritmos aún estaban ejecutando la última iteración.

**Tabla 10:** Comparación de los algoritmos considerando dos drones.

| Tipo | \|V\| | $\texttt{gap}_{\texttt{avg}}$ (%) | | | $\texttt{gap}_{\texttt{min}}$ (%) | | | $\texttt{hit}_{\texttt{avg}}^{\texttt{large}}$ | | | $\texttt{hit}_{\texttt{min}}^{\texttt{large}}$ | | | $\texttt{gap}_{\texttt{t}}$ (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** |
| **U** | **80** | 0,00 | 57,05 | 63,78 | 0,00 | 41,11 | 46,44 | 4 | 2 | 0 | 4 | 1 | 1 | 11,62 | 5,77 | 5833,41 |
| | **100** | 0,00 | 67,56 | 81,05 | 0,00 | 44,17 | 57,50 | 4 | 2 | 0 | 4 | 2 | 0 | 16,66 | 6,89 | 10868,61 |
| | **150** | 0,00 | 72,22 | 98,64 | 0,00 | 58,37 | 83,30 | 4 | 2 | 0 | 4 | 2 | 0 | 20,40 | 10,31 | 43191,60 |
| **S** | **40** | 0,00 | 41,36 | 44,64 | 0,00 | 24,02 | 28,29 | 4 | 2 | 0 | 4 | 2 | 0 | 7,55 | 2,95 | 1168,75 |
| | **60** | 0,00 | 49,65 | 55,77 | 0,00 | 28,83 | 30,17 | 4 | 2 | 0 | 4 | 2 | 0 | 10,94 | 4,28 | 2653,41 |
| | **80** | 0,00 | 71,02 | 75,81 | 0,00 | 56,59 | 58,86 | 4 | 2 | 0 | 4 | 2 | 0 | 10,88 | 5,63 | 6879,02 |
| **R** | **15** | 0,00 | 28,91 | 30,05 | 0,00 | 16,36 | 19,67 | 5 | 1 | 0 | 5 | 1 | 0 | 1,10 | 1,56 | 262,86 |
| | **30** | 0,00 | 41,96 | 47,17 | 0,00 | 26,44 | 30,50 | 4 | 2 | 0 | 4 | 1 | 1 | 3,70 | 2,68 | 772,29 |
| | **40** | 0,00 | 45,35 | 48,25 | 0,00 | 22,69 | 26,14 | 4 | 2 | 0 | 4 | 0 | 2 | 4,67 | 2,92 | 1436,07 |
| Promedio | | 0,00 | 52,79 | 60,57 | 0,00 | 35,40 | 42,32 | 4,11 | 1,89 | 0,00 | 4,11 | 1,44 | 0,44 | 9,73 | 4,78 | 8118,45 |
| **Suma de las 54 instancias** | | | | | | | | 37 | 17 | 0 | 37 | 13 | 4 | | | |

La Tabla 10 muestra que el VNS obtiene los mejores resultados, siendo el algoritmo base de comparación para la mayoría de las instancias. Obteniendo para el $\texttt{gap}_{\texttt{avg}}$ y el $\texttt{gap}_{\texttt{min}}$ un valor de $0,0\,\%$ cada uno, también se refleja en los 37 hits obtenidos. El SA es el segundo mejor en desempeño promedio, con un $\texttt{gap}_{\texttt{avg}}$ promedio de $52,79\,\%$ y un $\texttt{gap}_{\texttt{min}}$ promedio

de $35,40\,\%$. Nuevamente, el GA es el peor algoritmo con $\texttt{gap}_{\texttt{avg}}$ y $\texttt{gap}_{\texttt{min}}$ de $60,57\,\%$ y $42,32\,\%$, respectivamente. Esta vez, el GA logra cuatro hits. Las diferencias porcentuales entre el SA y el GA es de mayor magnitud que las diferencias obtenidas con un dron. Con respecto al tiempo límite, el SA es la heurística que presenta una menor diferencia porcentual para el $\texttt{gap}_{\texttt{t}}$, con un promedio del $4,78\,\%$. Nuevamente, el GA presenta una desviación más grande con el tiempo límite. El VNS tiene un promedio de ejecución que se desvía en promedio en un $9,73\,\%$.

**Tabla 11:** Comparación de los algoritmos considerando tres drones.

| Tipo | \|V\| | $\texttt{gap}_{\texttt{avg}}$ (%) | | | $\texttt{gap}_{\texttt{min}}$ (%) | | | $\texttt{hit}_{\texttt{avg}}^{\texttt{large}}$ | | | $\texttt{hit}_{\texttt{min}}^{\texttt{large}}$ | | | $\texttt{gap}_{\texttt{t}}$ (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** |
| **U** | **80** | 0,00 | 23,92 | 28,59 | 0,00 | 3,66 | 8,89 | 4 | 2 | 0 | 4 | 1 | 1 | 13,58 | 5,39 | 5108,49 |
| | **100** | 0,00 | 31,79 | 39,50 | 0,00 | 10,81 | 16,71 | 4 | 2 | 0 | 3 | 2 | 1 | 8,67 | 6,47 | 9707,60 |
| | **150** | 0,00 | 31,90 | 52,28 | 0,00 | 6,47 | 27,11 | 4 | 2 | 0 | 3 | 3 | 0 | 13,64 | 9,68 | 38656,19 |
| **S** | **40** | 0,00 | 21,63 | 25,77 | 0,00 | 4,35 | 8,06 | 4 | 2 | 0 | 4 | 2 | 0 | 6,66 | 2,79 | 912,63 |
| | **60** | 0,00 | 20,12 | 24,40 | 1,92 | 0,00 | 3,82 | 4 | 2 | 0 | 4 | 1 | 1 | 8,86 | 4,02 | 2713,36 |
| | **80** | 0,00 | 28,02 | 35,21 | 0,00 | 4,79 | 14,18 | 4 | 2 | 0 | 4 | 1 | 1 | 10,79 | 5,31 | 5455,09 |
| **R** | **15** | 0,00 | 12,05 | 16,89 | 0,00 | 6,64 | 9,08 | 4 | 2 | 0 | 3 | 2 | 1 | 1,37 | 1,48 | 179,21 |
| | **30** | 0,00 | 17,01 | 20,05 | 0,00 | 1,95 | 3,36 | 4 | 2 | 0 | 4 | 0 | 2 | 3,55 | 2,56 | 486,78 |
| | **40** | 0,00 | 21,50 | 25,79 | 0,00 | 1,75 | 5,92 | 4 | 2 | 0 | 3 | 2 | 1 | 4,77 | 2,77 | 890,39 |
| Promedio | | 0,00 | 23,11 | 29,83 | 0,21 | 4,49 | 10,79 | 4,00 | 2,00 | 0,00 | 3,56 | 1,56 | 0,89 | 7,99 | 4,50 | 7123,30 |
| **Suma de las 54 instancias** | | | | | | | | 36 | 18 | 0 | 32 | 14 | 8 | | | |

La Tabla 11 exhibe que el VNS obtiene los mejores resultados en promedio para el $\texttt{gap}_{\texttt{avg}}$ $(0,00\,\%)$ mientras que no siempre obtiene los mejores $\texttt{gap}_{\texttt{min}}$ $(0,21\,\%)$. Para este caso, el VNS obtiene peores resultados que el SA para el escenario Suburbano con 60 vértices, con una desviación del $1,92\,\%$. A pesar de esto, el VNS es el algoritmo con más hits, 36 y 32. El SA tiene un promedio para el $\texttt{gap}_{\texttt{avg}}$ y el $\texttt{gap}_{\texttt{min}}$ de $23,11\,\%$ y $4,49\,\%$, respectivamente. El GA obtiene un promedio para el $\texttt{gap}_{\texttt{avg}}$ de $29,83\,\%$ y para el $\texttt{gap}_{\texttt{min}}$ de $10,79\,\%$, siendo el algoritmo que peor se comporta considerando tres drones. El SA es la heurística que presenta una menor diferencia porcentual para el tiempo límite, con un promedio del $4,50\,\%$. El GA presenta la desviación más grande con el tiempo límite, con un $7123,30\,\%$. El VNS tiene un promedio de ejecución que se desvía en promedio en un $7,99\,\%$.

**Tabla 12:** Comparación de los algoritmos considerando cuatro drones.

| Tipo | \|V\| | $\texttt{gap}_{\texttt{avg}}$ (%) | | | $\texttt{gap}_{\texttt{min}}$ (%) | | | $\texttt{hit}_{\texttt{avg}}^{\texttt{large}}$ | | | $\texttt{hit}_{\texttt{min}}^{\texttt{large}}$ | | | $\texttt{gap}_{\texttt{t}}$ (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** |
| **U** | **80** | 0,00 | 3,73 | 10,01 | 10,54 | 0,00 | 6,17 | 3 | 3 | 0 | 1 | 4 | 1 | 11,66 | 5,25 | 4784,60 |
| | **100** | 0,00 | 3,16 | 10,74 | 11,24 | 0,00 | 6,71 | 3 | 3 | 0 | 2 | 2 | 2 | 19,33 | 6,29 | 8983,25 |
| | **150** | 0,00 | 2,68 | 23,88 | 10,56 | 0,00 | 15,78 | 4 | 2 | 0 | 1 | 4 | 1 | 27,49 | 9,41 | 37712,29 |
| **S** | **40** | 0,00 | 6,12 | 10,31 | 5,05 | 0,00 | 2,00 | 4 | 2 | 0 | 3 | 2 | 1 | 8,94 | 2,73 | 793,42 |
| | **60** | 0,00 | 0,81 | 6,84 | 10,12 | 0,00 | 2,61 | 4 | 2 | 0 | 1 | 2 | 3 | 11,98 | 3,90 | 2128,85 |
| | **80** | 0,00 | 7,24 | 15,67 | 7,27 | 0,00 | 6,03 | 4 | 2 | 0 | 2 | 4 | 0 | 13,38 | 5,17 | 5333,04 |
| **R** | **15** | 0,00 | 13,18 | 17,75 | 0,00 | 10,64 | 12,68 | 5 | 0 | 1 | 4 | 1 | 1 | 0,93 | 1,47 | 114,59 |
| | **30** | 0,00 | 2,95 | 8,78 | 0,00 | 0,64 | 3,06 | 4 | 2 | 0 | 4 | 1 | 1 | 3,10 | 2,52 | 483,82 |
| | **40** | 0,00 | 8,15 | 10,40 | 2,41 | 0,00 | 1,08 | 4 | 2 | 0 | 3 | 2 | 1 | 4,45 | 2,71 | 925,19 |
| Promedio | | 0,00 | 5,34 | 12,71 | 6,35 | 1,25 | 6,24 | 3,89 | 2,00 | 0,11 | 2,33 | 2,44 | 1,22 | 11,25 | 4,38 | 6806,56 |
| **Suma de las 54 instancias** | | | | | | | | 35 | 18 | 1 | 21 | 22 | 11 | | | |

La Tabla 12 muestra que el VNS obtiene los mejores resultados en promedio para el $\texttt{gap}_{\texttt{avg}}$, siendo la referencia para la comparación. VNS obtiene un promedio para el $\texttt{gap}_{\texttt{min}}$ de $6,35\,\%$, siendo referencia solamente para el escenario Rural con 15 y 30 vértices. Esto se refleja en la cantidad de hits obtenidos, con 35 y 21 para cada caso, respectivamente. El SA obtiene en promedio un $5,34\,\%$ y un $1,25\,\%$ para el $\texttt{gap}_{\texttt{avg}}$ y el $\texttt{gap}_{\texttt{min}}$. Comportándose de mejor manera al obtener resultados mínimos, esto se manifiesta en los 22 hits obtenidos. El GA obtiene un promedio para el $\texttt{gap}_{\texttt{avg}}$ de $12,71\,\%$ y para el $\texttt{gap}_{\texttt{min}}$ de $6,24\,\%$, para este último caso este algoritmo se comporta en promedio de mejor manera que el VNS. En cuanto al tiempo, el SA se mantiene el el rango de desviación de los anteriores casos con un $4,38\,\%$. El GA presenta la desviación del $6806,56\,\%$. Mientras que el VNS tiene un promedio de ejecución que se desvía en promedio en un $11,25\,\%$.

**Tabla 13:** Comparación de los algoritmos considerando cinco drones.

| Tipo | \|V\| | $\texttt{gap}_{\texttt{avg}}$ (%) | | | $\texttt{gap}_{\texttt{min}}$ (%) | | | $\texttt{hit}_{\texttt{avg}}^{\texttt{large}}$ | | | $\texttt{hit}_{\texttt{min}}^{\texttt{large}}$ | | | $\texttt{gap}_{\texttt{t}}$ (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** |
| **U** | **80** | 1,04 | 0,00 | 11,12 | 16,24 | 0,00 | 4,73 | 3 | 3 | 0 | 0 | 5 | 1 | 11,96 | 5,18 | 4765,46 |
| | **100** | 1,56 | 0,00 | 15,27 | 12,21 | 0,00 | 11,64 | 3 | 3 | 0 | 1 | 5 | 0 | 11,98 | 6,23 | 9064,74 |
| | **150** | 4,30 | 0,00 | 37,68 | 8,54 | 0,00 | 13,98 | 3 | 3 | 0 | 3 | 1 | 2 | 22,46 | 9,25 | 37464,93 |
| **S** | **40** | 0,00 | 3,80 | 8,32 | 2,12 | 0,00 | 3,09 | 4 | 2 | 0 | 3 | 3 | 1 | 7,58 | 2,73 | 864,69 |
| | **60** | 0,56 | 0,00 | 7,10 | 7,03 | 0,00 | 2,87 | 4 | 2 | 0 | 2 | 2 | 2 | 8,21 | 3,89 | 2110,85 |
| | **80** | 0,00 | 2,73 | 15,29 | 4,64 | 0,00 | 4,93 | 4 | 2 | 0 | 3 | 2 | 1 | 12,88 | 5,10 | 4960,09 |
| **R** | **15** | 0,00 | 14,02 | 16,90 | 0,00 | 16,71 | 18,87 | 6 | 0 | 0 | 6 | 0 | 0 | 0,78 | 1,47 | 180,89 |
| | **30** | 0,00 | 2,45 | 8,91 | 0,00 | 2,06 | 5,59 | 4 | 2 | 0 | 4 | 1 | 1 | 3,37 | 2,53 | 469,13 |
| | **40** | 0,00 | 3,50 | 8,88 | 0,47 | 0,40 | 0,00 | 4 | 2 | 0 | 3 | 2 | 1 | 5,17 | 2,71 | 867,83 |
| Promedio | | 0,83 | 2,94 | 14,39 | 5,69 | 2,13 | 7,30 | 3,89 | 2,11 | 0,00 | 2,78 | 2,33 | 1,00 | 9,38 | 4,34 | 6749,85 |
| **Suma de las 54 instancias** | | | | | | | | 35 | 19 | 0 | 25 | 21 | 9 | | | |

Por último, la Tabla 13 presenta que para el $\texttt{gap}_{\texttt{avg}}$, el VNS tiene un mejor rendimiento con un promedio del $0,83\,\%$. Por el lado del $\texttt{gap}_{\texttt{min}}$, el VNS tiene un promedio del $5,69\,\%$. Este algoritmo es el que más hits obtiene, con 35 y 25 para cada caso, respectivamente. Mientras que el SA es el algoritmo que obtienen los segundos mejores resultados en $\texttt{gap}_{\texttt{min}}$,

y es la referencia para la mayoría de los casos del $\texttt{gap}_{\texttt{min}}$, con un promedio del $2,13\,\%$. El GA no es la referencia de comparación para ningún caso, obteniendo un promedio de $14,39\,\%$ y $7,30\,\%$ para el $\texttt{gap}_{\texttt{avg}}$ y el $\texttt{gap}_{\texttt{min}}$, respectivamente. Nuevamente en el tiempo, el SA es el algoritmo que menos se desvía de los 10 segundo, con un $4,34\,\%$. El GA sobrepasa el límite en un promedio del $6749,85\,\%$. Mientras que el VNS tiene una diferencia porcentual promedio del $9,38\,\%$.

**Tabla 14:** Resumen de la comparación entre algoritmos.

| $m$ | Promedio $\texttt{gap}_{\texttt{avg}}$ (%) | | | Promedio $\texttt{gap}_{\texttt{min}}$ (%) | | | Total $\texttt{hit}^{\texttt{large}}_{\texttt{avg}}$ | | | Total $\texttt{hit}^{\texttt{large}}_{\texttt{min}}$ | | | Promedio $\texttt{gap}_{\texttt{t}}$ (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** | **VNS** | **SA** | **GA** |
| 1 | 0,00 | 8,64 | 22,78 | 0,00 | 6,26 | 16,84 | 43 | 11 | 0 | 40 | 12 | 2 | 8,05 | 5,26 | 9769,60 |
| 2 | 0,00 | 52,79 | 60,57 | 0,00 | 35,40 | 42,32 | 37 | 17 | 0 | 37 | 13 | 4 | 9,73 | 4,78 | 8118,45 |
| 3 | 0,00 | 23,11 | 29,83 | 0,21 | 4,49 | 10,79 | 36 | 18 | 0 | 32 | 14 | 8 | 7,99 | 4,50 | 7123,30 |
| 4 | 0,00 | 5,34 | 12,71 | 6,35 | 1,25 | 6,24 | 35 | 18 | 1 | 21 | 22 | 11 | 11,25 | 4,38 | 6806,56 |
| 5 | 0,83 | 2,94 | 14,39 | 5,69 | 2,13 | 7,30 | 35 | 19 | 0 | 25 | 21 | 9 | 9,38 | 4,34 | 6749,85 |
| Promedio | 0,17 | 18,56 | 28,06 | 2,45 | 9,91 | 16,70 | 37,20 | 16,60 | 0,20 | 31,00 | 16,40 | 6,80 | 9,28 | 4,65 | 7713,55 |

Las métricas expuestas anteriormente se presentan resumidas en la Tabla 14. A modo general, el VNS obtiene una mejor media de los resultados reflejado en el promedio para $\texttt{gap}_{\texttt{avg}}$, de $0,17\,\%$. Sin embargo, no sucede lo mismo en los resultados mínimos resueltos con todas las cantidades de drones. Si se observan los resultados con uno, dos y tres drones, el VNS se comporta de mejor manera, obteniendo un mejor promedio para el $\texttt{gap}_{\texttt{min}}$. Mientras que para una mayor cantidad de drones, el SA se comporta de mejor manera obteniendo el mejores promedios para $\texttt{gap}_{\texttt{min}}$, con un promedio del $1,25\,\%$ considerando cuatro drones y $2,13\,\%$ considerando cinco drones. En cuanto a la cantidad de hits obtenidos, el VNS es el algoritmo con mejor rendimiento, con un promedio de 37,20 y 31,00 para $\texttt{hit}^{\texttt{large}}_{\texttt{avg}}$ y $\texttt{hit}^{\texttt{large}}_{\texttt{min}}$, respectivamente. Siendo superado por el SA, por una unidad, cuando se considera el uso de cuatro drones. El GA es el algoritmo de peor rendimiento, reflejándose en la pequeña cantidad de hits alcanzados. En resumen, VNS logra obtener las mejores soluciones en 186 de las 270 instancias en $\texttt{hit}^{\texttt{large}}_{\texttt{avg}}$ y 155 de las 270 instancias en $\texttt{hit}^{\texttt{large}}_{\texttt{min}}$.

Con respecto al tiempo límite, el algoritmo SA es el que menor desviación porcentual tiene. Con un mínimo de $4,34\,\%$ para el uso de cinco drones y un máximo de $5,26\,\%$ con la utilización de un dron. Mientras que el VNS tiene un $\texttt{gap}_{\texttt{t}}$ mínimo de un $8,05\,\%$ y un máximo de $11,25\,\%$, considerando la utilización de uno y cuatro drones respectivamente. El GA es el algoritmo que tiene en promedio la ejecución más lenta, con un promedio $\texttt{gap}_{\texttt{t}}$ de $7713,55\,\%$. Una posible suposición de la razón de que el VNS obtiene mejores soluciones en el uso de cantidades reducidas de drones, es la forma en que operan las búsquedas locales y perturbaciones. Donde se evalúa un dron por iteración, limitando a

que los vecindarios analizados sean proporcional la cantidad de iteraciones y al tiempo límite establecido.

**Tabla 15:** Resumen de la prueba de Wilcoxon.

| | **Diferencia VNS–SA** | **Diferencia VNS–GA** |
|---|---|---|
| **Estadístico Wilcoxon** | 14654 | 10625 |
| **Valor-$p$** | 0,022 | $1,2128 \times 10^{-5}$ |

Finalmente, se realiza una prueba de Wilcoxon (Carrasco *et al.*, 2020), con el fin de verificar estadísticamente que los resultados del VNS tienen una diferencia con los resultados de los otros algoritmos. Esta prueba no paramétrica es análoga a la prueba t de Student, caracterizada por presentar una mayor sensibilidad (Derrac *et al.*, 2011). Para un nivel de significancia del 95 %, se seleccionan los costos mínimos obtenidos por los algoritmos, teniendo un tamaño de 270 correspondiente a las instancias para el *Set large*. Obteniendo un valor–$p$ para las diferencias de 0,0226 entre el VNS y SA y $1,2128 \times 10^{-5}$ entre el VNS y GA. Indicando que para las dos comparaciones se rechaza la hipótesis nula de que la mediana de las diferencias es igual a cero, verificando que existe diferencia estadísticamente significativa entre los resultados. La Tabla 15 presenta los estadísticos calculados y los respectivos valores–$p$ para las comparaciones.

### 5.5.2. Resultados del *Set small*

Las Tablas 16–20 presentan la comparación de los resultados del modelo de MILP, el VNS, SA y GA para las instancias del *Set small* considerando uno, dos, tres, cuatro y cinco drones, respectivamente. Las tablas mencionadas en esta sección tienen el siguiente formato: la primera columna se refiere a el tamaño del plano para los tipos de escenario Urbano (U), Suburbano (S) y Rural (R). La segunda columna indica la cantidad de vértices presentes en la instancia. La tercera, cuarta y quinta columna corresponden al límite superior, límite inferior y el tiempo de ejecución del modelo. Mientras que la sexta columna reporta el gap entre los límites inferior y superior ($\texttt{gap}_{\texttt{MILP}}$). Luego, la séptima, octava, novena y décima columna presentan la medida de calidad $\texttt{gap}_{\texttt{UB}}$ para el modelo de MILP, VNS, SA y el GA, respectivamente. A continuación, las siguientes cuatro columnas presentan la cantidad de hits () que obtienen el MILP, VNS, SA y GA. Finalmente, la penúltima fila muestra el promedio de las columnas del gap calculado para el modelo y cada algoritmo. Mientras que la última fila muestra la totalidad de hits obtenidos, teniendo un máximo de 108 por la cantidad de instancias que son resueltas considerando un número determinado de drones.

**Tabla 16:** Comparación de modelo de MILP y los algoritmos considerando un dron.

| Tipo | \|V\| | UB | LB | Tiempo | $\texttt{gap}_{\texttt{MILP}}$ (%) | $\texttt{gap}_{\texttt{UB}}$ (%) | | | | $\texttt{hit}^{\texttt{small}}_{\texttt{min}}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | MILP | VNS | SA | GA | MILP | VNS | SA | GA |
| U | **5** | 23,01 | 23,01 | 0,1 | 0,00 | 0,00 | 16,61 | 87,33 | 87,04 | 6 | 4 | 0 | 0 |
| | **8** | 27,22 | 27,22 | 32,4 | 0,00 | 0,00 | 21,57 | 58,36 | 64,89 | 6 | 1 | 1 | 1 |
| | **12** | 39,80 | 4,75 | 3600,0 | 816,33 | 0,00 | 22,67 | 46,08 | 54,54 | 6 | 2 | 0 | 0 |
| | **15** | 57,18 | 3,41 | 3600,1 | 1758,38 | 0,82 | 7,30 | 19,82 | 25,32 | 6 | 4 | 2 | 2 |
| | **18** | 62,89 | 3,27 | 3600,4 | 3287,33 | 5,45 | 5,60 | 12,94 | 16,32 | 4 | 5 | 4 | 3 |
| | **20** | 77,97 | 2,91 | 3600,0 | 2965,41 | 15,45 | 1,91 | 7,50 | 8,16 | 2 | 6 | 4 | 3 |
| S | **5** | 44,23 | 44,23 | 0,1 | 0,00 | 0,00 | 29,44 | 82,88 | 86,54 | 6 | 4 | 1 | 0 |
| | **8** | 55,07 | 55,06 | 21,6 | 0,00 | 0,00 | 26,53 | 71,19 | 74,09 | 6 | 2 | 0 | 0 |
| | **12** | 74,36 | 9,25 | 3600,0 | 864,37 | 0,00 | 22,69 | 41,01 | 45,10 | 6 | 1 | 0 | 0 |
| | **15** | 105,01 | 5,08 | 3600,0 | 2421,77 | 2,61 | 5,74 | 14,74 | 16,87 | 5 | 5 | 3 | 1 |
| | **18** | 111,05 | 5,30 | 3600,0 | 2496,57 | 4,27 | 3,06 | 17,38 | 18,56 | 5 | 6 | 2 | 1 |
| | **20** | 154,31 | 5,61 | 3600,0 | 4325,11 | 16,17 | 1,66 | 5,19 | 6,31 | 3 | 6 | 5 | 5 |
| R | **5** | 150,89 | 150,89 | 0,0 | 0,00 | 3,18 | 0,04 | 19,34 | 20,91 | 5 | 6 | 3 | 3 |
| | **8** | 132,67 | 132,67 | 17,4 | 0,00 | 0,00 | 12,31 | 36,11 | 38,40 | 6 | 3 | 2 | 1 |
| | **12** | 158,16 | 12,75 | 3600,0 | 2341,10 | 0,00 | 18,83 | 30,82 | 31,38 | 6 | 3 | 1 | 1 |
| | **15** | 203,15 | 13,87 | 3600,0 | 1743,11 | 5,07 | 13,15 | 21,56 | 21,30 | 5 | 3 | 2 | 2 |
| | **18** | 250,72 | 13,58 | 3600,0 | 2402,81 | 8,94 | 3,26 | 11,90 | 13,44 | 4 | 5 | 3 | 2 |
| | **20** | 283,48 | 10,60 | 3600,0 | 7011,58 | 20,45 | 4,14 | 12,83 | 13,66 | 2 | 5 | 4 | 2 |
| Promedio | | | | | 1801,88 | 4,58 | 12,03 | 33,17 | 35,71 | 4,94 | 3,94 | 2,06 | 1,50 |
| **Suma de las 108 instancias** | | | | | | | | | | 89 | 71 | 37 | 27 |

La Tabla 16 muestra que el modelo de MILP puede demostrar optimalidad para las instancias 5 y 8 drones ($\texttt{gap}_{\texttt{MILP}}$ igual a cero). Mientras que para las instancias con mayor cantidad de vértices, el modelo obtiene un gap entre los límites de mayor magnitud. Esto se ve reflejado en el tiempo de ejecución requerido, donde en las instancias pequeñas se requieren menos de una hora de ejecución. Por el lado de $\texttt{gap}_{\texttt{UB}}$, el modelo obtiene el promedio más bajo con un $4,58\,\%$ y el VNS con un $12,03\,\%$ es el algoritmo con el menor promedio. Indicando que es el algoritmo que mejor resultados obtiene, en general, tiene un gap de menor magnitud para las instancias con 15, 18 y 20 vértices. El SA y el GA tienen un promedio del $\texttt{gap}_{\texttt{UB}}$ similar, además de seguir la tendencia de tener una magnitud de la diferencia porcentual con el $UB$ de mayor magnitud para instancias con poca cantidad de vértices. El VNS es el algoritmo que más hits obtiene con 71, seguido por el SA con 37 y el GA con 27. Sin embargo el modelo obtiene 89 hits, siendo la mayor cantidad. Cabe destacar que se aprecia que a medida que el número de vértices aumenta, el rendimiento del modelo empeora. Pero, el rendimiento del VNS mejora considerablemente, en menor medida lo hacen SA y GA.

**Tabla 17:** Comparación de modelo de MILP y los algoritmos considerando dos drones.

| Tipo | \|V\| | UB | LB | Tiempo | $\texttt{gap}_{\texttt{MILP}}$ (%) | $\texttt{gap}_{\texttt{UB}}$ (%) | | | | $\texttt{hit}^{\texttt{small}}_{\texttt{min}}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | MILP | VNS | SA | GA | MILP | VNS | SA | GA |
| U | **5** | 18,07 | 18,07 | 0,1 | 0,00 | 0,00 | 0,00 | 109,25 | 109,40 | 6 | 6 | 0 | 0 |
| | **8** | 22,94 | 22,94 | 21,8 | 0,00 | 1,54 | 18,55 | 84,91 | 93,99 | 6 | 2 | 0 | 0 |
| | **12** | 30,80 | 4,73 | 3600,0 | 680,52 | 0,00 | 30,55 | 79,55 | 82,93 | 6 | 1 | 0 | 0 |
| | **15** | 52,05 | 3,96 | 3600,0 | 1379,71 | 3,44 | 4,74 | 23,23 | 30,90 | 5 | 4 | 2 | 2 |
| | **18** | 54,33 | 3,29 | 3600,6 | 2414,04 | 4,22 | 14,85 | 24,77 | 26,87 | 4 | 2 | 2 | 2 |
| | **20** | 68,20 | 2,97 | 3600,0 | 2451,53 | 12,54 | 8,50 | 19,02 | 19,99 | 4 | 4 | 2 | 2 |
| S | **5** | 36,14 | 36,14 | 0,1 | 0,00 | 0,00 | 16,69 | 113,14 | 115,87 | 6 | 5 | 1 | 0 |
| | **8** | 46,34 | 46,34 | 17,1 | 0,00 | 0,00 | 18,19 | 96,10 | 98,37 | 6 | 2 | 0 | 0 |
| | **12** | 55,39 | 9,51 | 3600,0 | 593,19 | 0,00 | 48,91 | 80,94 | 84,88 | 6 | 0 | 0 | 0 |
| | **15** | 86,93 | 5,48 | 3600,1 | 1824,17 | 2,04 | 17,49 | 32,93 | 35,24 | 5 | 1 | 0 | 1 |
| | **18** | 107,59 | 5,31 | 3600,2 | 2153,39 | 10,92 | 11,06 | 42,11 | 40,98 | 4 | 4 | 1 | 1 |
| | **20** | 126,77 | 5,61 | 3600,0 | 3579,11 | 4,16 | 3,71 | 11,51 | 16,93 | 5 | 6 | 3 | 2 |
| R | **5** | 138,96 | 138,96 | 0,0 | 0,00 | 2,03 | 0,04 | 28,07 | 29,73 | 5 | 6 | 3 | 3 |
| | **8** | 125,85 | 125,85 | 12,9 | 0,00 | 0,05 | 6,44 | 29,03 | 28,69 | 6 | 5 | 2 | 1 |
| | **12** | 147,23 | 12,10 | 3600,0 | 2245,95 | 0,00 | 20,38 | 42,09 | 43,09 | 6 | 2 | 0 | 0 |
| | **15** | 175,85 | 15,02 | 3600,2 | 1100,17 | 4,62 | 29,62 | 45,76 | 49,56 | 5 | 4 | 1 | 1 |
| | **18** | 209,23 | 13,42 | 3600,0 | 2049,07 | 2,25 | 11,32 | 29,78 | 34,75 | 5 | 4 | 0 | 1 |
| | **20** | 272,16 | 10,63 | 3600,0 | 6623,30 | 21,65 | 6,68 | 16,60 | 15,57 | 2 | 4 | 3 | 4 |
| Promedio | | | | | 1505,23 | 3,86 | 14,87 | 50,49 | 53,21 | 5,11 | 3,44 | 1,11 | 1,11 |
| **Suma de las 108 instancias** | | | | | | | | | | 92 | 62 | 20 | 20 |

La Tabla 17 ilustra que el modelo obtiene un $\texttt{gap}_{\texttt{MILP}}$ igual a cero, es decir, soluciones óptimas para las instancias 5 y 8 drones. Para las instancias con mayor cantidad de vértices, es decir con 12, 15, 18 o 20, el modelo obtiene un gap entre límites de mayor magnitud. El tiempo de ejecución requerido también es mayor en este grupo de instancias más grandes, requiriendo una hora de ejecución para obtener soluciones factibles. Observando la métrica $\texttt{gap}_{\texttt{UB}}$, el modelo obtienen el menor promedio, con un $3,86\,\%$. Respecto a los algoritmos, el VNS obtiene el promedio más bajo con un $14,87\,\%$. En esta ocasión, el VNS obtiene un gap de menor magnitud para las instancias con 5, 18 y 20 de cada tipo de escenario. Mientras que el SA, con un $\texttt{gap}_{\texttt{UB}}$ promedio de $50,49\,\%$, obtienen una mayor diferencia con el $UB$ para instancias pequeñas, reduciendo esta diferencia si se evalúan instancias con 15 o más vértices. El GA sigue la misma tendencia del SA, pero con una leve mayor magnitud de los gap. Esto ultimo se aprecia en el $\texttt{gap}_{\texttt{UB}}$ de $53,21\,\%$. El modelo obtiene la mayor cantidad de hits, con 92, seguido por el VNS con 62, el SA y el GA con 20 cada uno.

**Tabla 18:** Comparación de modelo de MILP y los algoritmos considerando tres drones.

| Tipo | \|V\| | UB | LB | Tiempo | $\texttt{gap}_{\texttt{MILP}}$ (%) | $\texttt{gap}_{\texttt{UB}}$ (%) | | | | $\texttt{hit}^{\texttt{small}}_{\texttt{min}}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | MILP | VNS | SA | GA | MILP | VNS | SA | GA |
| U | **5** | 15,58 | 15,58 | 0,1 | 0,00 | 0,00 | 0,00 | 139,14 | 139,34 | 6 | 6 | 0 | 0 |
| | **8** | 19,53 | 19,53 | 19,4 | 0,00 | 0,00 | 10,58 | 101,76 | 108,16 | 6 | 3 | 0 | 0 |
| | **12** | 26,01 | 5,08 | 3600,0 | 467,98 | 0,00 | 43,55 | 111,19 | 112,37 | 6 | 0 | 0 | 0 |
| | **15** | 46,14 | 3,94 | 3600,0 | 1220,40 | 4,50 | 11,30 | 35,02 | 41,16 | 4 | 3 | 1 | 1 |
| | **18** | 59,32 | 3,58 | 3600,0 | 2851,85 | 17,36 | 11,01 | 19,45 | 20,56 | 4 | 3 | 2 | 2 |
| | **20** | 58,51 | 3,06 | 3600,0 | 1941,91 | 7,36 | 15,35 | 17,67 | 21,49 | 5 | 2 | 2 | 2 |
| S | **5** | 30,96 | 30,96 | 0,1 | 0,00 | 0,00 | 17,23 | 414,58 | 417,32 | 6 | 5 | 1 | 0 |
| | **8** | 40,67 | 40,67 | 14,5 | 0,00 | 0,00 | 23,89 | 131,84 | 140,14 | 6 | 2 | 0 | 0 |
| | **12** | 47,46 | 9,55 | 3600,0 | 486,85 | 0,00 | 38,95 | 104,75 | 108,27 | 6 | 2 | 0 | 0 |
| | **15** | 80,20 | 5,46 | 3600,0 | 1706,07 | 1,84 | 22,26 | 34,24 | 35,85 | 5 | 2 | 1 | 1 |
| | **18** | 98,28 | 5,76 | 3600,1 | 1929,69 | 11,30 | 14,66 | 39,33 | 40,26 | 5 | 3 | 1 | 1 |
| | **20** | 135,66 | 5,72 | 3600,3 | 2942,21 | 20,50 | 9,04 | 15,32 | 17,51 | 4 | 3 | 3 | 3 |
| R | **5** | 134,21 | 134,21 | 0,0 | 0,00 | 2,03 | 0,04 | 75,97 | 76,97 | 5 | 6 | 3 | 3 |
| | **8** | 119,02 | 119,02 | 11,5 | 0,00 | 1,08 | 2,40 | 56,72 | 56,27 | 6 | 6 | 2 | 1 |
| | **12** | 132,30 | 11,87 | 3600,0 | 2078,68 | 0,09 | 30,42 | 53,30 | 52,93 | 6 | 2 | 1 | 1 |
| | **15** | 168,77 | 14,82 | 3600,0 | 1107,45 | 3,93 | 20,42 | 28,78 | 32,57 | 5 | 3 | 1 | 1 |
| | **18** | 178,70 | 12,63 | 3600,0 | 1680,41 | 0,00 | 28,29 | 35,89 | 42,39 | 6 | 1 | 1 | 1 |
| | **20** | 225,00 | 11,16 | 3600,0 | 5505,57 | 11,86 | 15,36 | 8,03 | 7,45 | 3 | 0 | 4 | 4 |
| Promedio | | | | | 1328,84 | 4,55 | 17,49 | 79,05 | 81,72 | 5,22 | 2,89 | 1,28 | 1,17 |
| **Suma de las 108 instancias** | | | | | | | | | | 94 | 52 | 23 | 21 |

La Tabla 18 nuevamente presenta que el modelo obtiene un $\texttt{gap}_{\texttt{MILP}}$ igual a cero, para las instancias 5 y 8 drones. Para las instancias con mayor cantidad de vértices, el modelo obtiene un gap entre límites de mayor magnitud que los casos analizados anteriormente. El tiempo de ejecución requerido para las instancias con 12 o más vértices se mantiene en una hora para encontrar soluciones factibles. Nuevamente el modelo de MILP obtiene el mejor promedio para $\texttt{gap}_{\texttt{UB}}$. Mientras, VNS logra alcanzar el promedio más bajo de la diferencia con respecto al $UB$, con un $17,49\,\%$. Los gap de menor valor se encuentran en las instancias con 5, 18 y 20 vértices. En contraste, el SA y el GA con un promedio de $79,05\,\%$ y $81,72\,\%$, respectivamente, siguen con la tendencia de obtener peores resultados en instancias con poca cantidad de vértices. No obstante, la magnitud de estas diferencias es notoriamente mayor a las instancias resueltas con menor cantidad de drones. El MILP obtiene 94 hits, seguido por el VNS como el algoritmo con mayor cantidad de hits. El SA y GA obtienen 23 y 21 hits, respectivamente.

**Tabla 19:** Comparación de modelo de MILP y los algoritmos considerando cuatro drones.

| Tipo | \|V\| | UB | LB | Tiempo | $\texttt{gap}_{\texttt{MILP}}$ (%) | $\texttt{gap}_{\texttt{UB}}$ (%) | | | | $\texttt{hit}^{\texttt{small}}_{\texttt{min}}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | MILP | VNS | SA | GA | MILP | VNS | SA | GA |
| U | **5** | 15,58 | 15,58 | 0,1 | 0,00 | 0,00 | 0,00 | 139,14 | 139,34 | 6 | 6 | 0 | 0 |
| | **8** | 19,12 | 19,12 | 21,0 | 0,00 | 0,00 | 4,03 | 102,80 | 109,58 | 6 | 5 | 0 | 0 |
| | **12** | 24,67 | 5,09 | 3600,0 | 425,49 | 0,00 | 35,29 | 116,43 | 120,43 | 6 | 1 | 0 | 0 |
| | **15** | 43,02 | 3,79 | 3600,1 | 1193,89 | 1,20 | 13,51 | 42,66 | 51,21 | 6 | 3 | 1 | 1 |
| | **18** | 54,42 | 3,37 | 3600,3 | 2776,19 | 14,98 | 15,23 | 29,76 | 29,74 | 4 | 1 | 2 | 2 |
| | **20** | 58,23 | 3,04 | 3600,0 | 2026,47 | 5,78 | 10,24 | 14,32 | 20,46 | 5 | 4 | 3 | 2 |
| S | **5** | 30,96 | 30,96 | 0,1 | 0,00 | 0,00 | 17,23 | 414,58 | 417,32 | 6 | 5 | 1 | 0 |
| | **8** | 39,57 | 39,57 | 13,7 | 0,00 | 0,00 | 11,85 | 144,22 | 153,41 | 6 | 3 | 0 | 0 |
| | **12** | 48,72 | 9,87 | 3600,0 | 516,60 | 1,99 | 18,66 | 104,64 | 110,36 | 5 | 3 | 0 | 0 |
| | **15** | 81,81 | 5,65 | 3600,1 | 1747,85 | 8,96 | 14,82 | 43,21 | 44,79 | 4 | 2 | 1 | 1 |
| | **18** | 98,71 | 5,19 | 3600,3 | 1928,16 | 11,32 | 11,31 | 38,83 | 38,45 | 5 | 5 | 2 | 1 |
| | **20** | 123,75 | 5,23 | 3600,0 | 2861,59 | 13,69 | 7,05 | 16,68 | 16,54 | 4 | 4 | 2 | 3 |
| R | **5** | 134,21 | 134,21 | 0,0 | 0,00 | 2,03 | 0,04 | 75,97 | 76,97 | 5 | 6 | 3 | 3 |
| | **8** | 117,38 | 117,38 | 11,0 | 0,00 | 1,67 | 1,67 | 68,73 | 70,37 | 6 | 6 | 2 | 1 |
| | **12** | 142,47 | 12,10 | 3600,0 | 2203,57 | 4,37 | 13,64 | 44,48 | 46,08 | 5 | 4 | 1 | 1 |
| | **15** | 151,64 | 13,46 | 3600,0 | 1079,11 | 1,45 | 30,11 | 43,73 | 47,16 | 6 | 2 | 0 | 0 |
| | **18** | 197,43 | 14,44 | 3600,2 | 1693,86 | 5,91 | 19,66 | 26,76 | 31,50 | 5 | 2 | 1 | 0 |
| | **20** | 240,63 | 11,47 | 3600,0 | 5369,64 | 15,44 | 12,30 | 10,89 | 13,10 | 3 | 2 | 4 | 3 |
| Promedio | | | | | 1323,47 | 4,93 | 13,15 | 82,10 | 85,38 | 5,17 | 3,56 | 1,28 | 1,00 |
| **Suma de las 108 instancias** | | | | | | | | | | 93 | 64 | 23 | 18 |

La Tabla 19 muestra que el modelo continúa obteniendo soluciones óptimas para instancias con 5 y 8 vértices, mientras que las para las demás instancias requiere una hora de ejecución para obtener soluciones factibles. El $\texttt{gap}_{\texttt{MILP}}$ de estas soluciones se incrementa a medida que se considera un mayor número de clientes. El modelo de MILP tiene un promedio de $4,93\,\%$ para $\texttt{gap}_{\texttt{UB}}$. Por parte de los algoritmos, el VNS mantiene el menor promedio para el $\texttt{gap}_{\texttt{UB}}$ con un $13,15\,\%$. Mientras que el SA tiene en promedio un $82,10\,\%$ y el GA un $85,38\,\%$. La magnitud de los gap conseguidos por estos últimos dos algoritmos disminuye a medida que se abordan instancias con mayor cantidad de clientes y se cambia de tipo de escenario a uno suburbano o rural. El modelo de MILP obtiene 93 hits, el VNS consigue 64 hits, el SA obtiene 23 y el GA 18.

**Tabla 20:** Comparación de modelo de MILP y los algoritmos considerando cinco drones.

| Tipo | \|V\| | UB | LB | Tiempo | $\texttt{gap}_{\texttt{MILP}}$ (%) | $\texttt{gap}_{\texttt{UB}}$ (%) | | | | $\texttt{hit}^{\texttt{small}}_{\texttt{min}}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | MILP | VNS | SA | GA | MILP | VNS | SA | GA |
| U | **5** | 15,58 | 15,58 | 0,1 | 0,00 | 0,00 | 0,00 | 139,14 | 139,34 | 6 | 6 | 0 | 0 |
| | **8** | 18,16 | 18,16 | 16,4 | 0,00 | 0,00 | 3,19 | 116,02 | 123,45 | 6 | 6 | 0 | 0 |
| | **12** | 23,64 | 5,51 | 3600,0 | 417,67 | 4,16 | 23,77 | 132,19 | 141,17 | 5 | 3 | 0 | 0 |
| | **15** | 42,71 | 3,96 | 3600,0 | 1146,23 | 3,61 | 12,65 | 48,37 | 54,50 | 5 | 4 | 1 | 1 |
| | **18** | 51,10 | 3,43 | 3600,0 | 2334,31 | 8,72 | 9,99 | 29,85 | 32,55 | 4 | 3 | 2 | 2 |
| | **20** | 57,31 | 2,95 | 3600,0 | 2044,58 | 11,58 | 12,71 | 23,37 | 27,98 | 5 | 4 | 3 | 2 |
| S | **5** | 30,96 | 30,96 | 0,1 | 0,00 | 0,00 | 17,23 | 414,58 | 417,32 | 6 | 5 | 1 | 0 |
| | **8** | 38,34 | 38,34 | 14,5 | 0,00 | 0,00 | 7,22 | 161,55 | 171,24 | 6 | 4 | 0 | 0 |
| | **12** | 36,79 | 9,79 | 3600,2 | 344,13 | 1,30 | 43,58 | 169,70 | 179,71 | 6 | 2 | 0 | 0 |
| | **15** | 70,48 | 5,45 | 3600,1 | 1419,41 | 3,16 | 12,47 | 61,11 | 62,87 | 5 | 4 | 0 | 0 |
| | **18** | 94,40 | 5,42 | 3600,0 | 1740,45 | 12,40 | 7,10 | 40,14 | 40,74 | 5 | 4 | 1 | 1 |
| | **20** | 126,39 | 5,76 | 3600,4 | 2835,48 | 20,93 | 10,07 | 21,64 | 23,48 | 3 | 5 | 2 | 2 |
| R | **5** | 134,21 | 134,21 | 0,0 | 0,00 | 2,03 | 0,04 | 75,97 | 76,97 | 5 | 6 | 3 | 3 |
| | **8** | 116,25 | 116,25 | 11,3 | 0,00 | 1,08 | 1,78 | 77,78 | 81,69 | 6 | 6 | 2 | 1 |
| | **12** | 126,71 | 13,33 | 3600,0 | 1925,78 | 0,54 | 23,60 | 74,04 | 76,06 | 6 | 3 | 1 | 1 |
| | **15** | 166,58 | 13,04 | 3600,0 | 1275,67 | 7,73 | 23,78 | 45,03 | 47,98 | 5 | 3 | 1 | 1 |
| | **18** | 179,32 | 13,47 | 3600,0 | 1732,16 | 2,87 | 11,77 | 32,02 | 41,44 | 5 | 3 | 0 | 0 |
| | **20** | 232,77 | 11,19 | 3600,0 | 5453,73 | 10,87 | 4,13 | 8,84 | 9,09 | 5 | 5 | 5 | 5 |
| Promedio | | | | | 1259,42 | 5,05 | 12,50 | 92,85 | 97,09 | 5,22 | 4,22 | 1,22 | 1,06 |
| **Suma de las 108 instancias** | | | | | | | | | | 94 | 76 | 22 | 19 |

La Tabla 20 muestra que el modelo de MILP obtiene un límite superior igual al inferior para las instancias 5 y 8 drones. Mientras que para las instancias con mayor cantidad de vértices, el modelo obtiene un gap de mayor magnitud teniendo una diferencia en las instancias con 18 y 20 vértices. La obtención de soluciones factibles, y no óptimas, coincide en la hora de ejecución requerida. EL modelo de MILP se mantiene con el mejor desempeño, con un promedio para $\texttt{gap}_{\texttt{UB}}$ de un $5,05\,\%$. Por el lado de los algoritmos, el VNS obtiene el promedio más bajo con un $12,50\,\%$. Los gap con menor magnitud los consigue en las instancias con 5, 18 y 20 vértices. El SA y el GA tienen un promedio del $\texttt{gap}_{\texttt{UB}}$ de $92,85\,\%$ y $97,09\,\%$. Con la tendencia, al igual que en los casos anteriormente expuestos, de obtener un gap de mayor magnitud si se evalúan instancias con menor cantidad de clientes. Nuevamente, el MILP obtiene la mayor cantidad de hots. El VNS obtiene 71 hits, el SA 37, por último el GA alcanza 18.

**Tabla 21:** Resumen de la comparación entre el modelo de MILP y los algoritmos.

| $m$ | Promedio $\texttt{gap}_{\texttt{MILP}}$ (%) | Promedio $\texttt{gap}_{\texttt{UB}}$ (%) | | | | Total $\texttt{hit}^{\texttt{small}}_{\texttt{min}}$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | MILP | VNS | SA | GA | Modelo | VNS | SA | GA |
| **1** | 1801,88 | 4,58 | 12,03 | 33,17 | 35,71 | 89,00 | 71,00 | 37,00 | 27,00 |
| **2** | 1505,23 | 3,86 | 14,87 | 50,49 | 53,21 | 92,00 | 62,00 | 20,00 | 20,00 |
| **3** | 1328,84 | 4,55 | 17,49 | 79,05 | 81,72 | 94,00 | 52,00 | 23,00 | 21,00 |
| **4** | 1328,84 | 4,55 | 17,49 | 79,05 | 81,72 | 93,00 | 64,00 | 23,00 | 18,00 |
| **5** | 1259,42 | 5,05 | 12,50 | 92,85 | 97,09 | 92,00 | 62,25 | 25,75 | 21,50 |
| Promedio | 1444,84 | 4,52 | 14,88 | 66,92 | 69,89 | 92,00 | 62,25 | 25,75 | 21,50 |

La Tabla 21 ilustra el resumen de la descripción realizada para los casos con distinta cantidad de drones. A modo general, para la mayoría de las instancias el modelo de MILP obtiene mejores resultados que los algoritmos a pesar de que no se obtengan óptimos, demostrado con un promedio de $4,52\,\%$. Sin embargo, en la mayoría de las instancias se obtienen soluciones factibles en una hora de ejecución, a diferencia de los algoritmos. Esto se refleja en el promedio de $1444,84\,\%$ para $\texttt{gap}_{\texttt{MILP}}$. Es importante mencionar que el modelo obtuvo soluciones óptimas para las instancias con 5 y 8 vértices, consiguiendo la misma cantidad de óptimos considerando todas las cantidades de drones. En tanto que el VNS es el algoritmo que en promedio obtiene una menor diferencia con el mejor resultado obtenido, esto se ve reflejado de igual manera en los 92 hits obtenidos en total. Por el lado del SA y GA, estos tienen gap que están en el mismo rango porcentual para la misma cantidad de drones utilizados. Una hipótesis a plantear es que las propuestas de estos dos algoritmos presentados en la literatura, no logran acercarse al óptimo frecuentemente, quedando atrapados al encontrar soluciones óptimas locales.

**Tabla 22:** Resumen de la comparación entre el modelo de MILP y los algoritmos.

| **Tipo** | **\|V\|** | Promedio $\texttt{gap}_{\texttt{MILP}}$ (%) | Promedio $\texttt{gap}_{\texttt{UB}}$ (%) | | | | Total $\texttt{hit}^{\texttt{small}}_{\texttt{min}}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | **MILP** | **VNS** | **SA** | **GA** | **Modelo** | **VNS** | **SA** | **GA** |
| **U** | **5** | 0,00 | 0,00 | 3,32 | 122,80 | 122,89 | 6,00 | 5,60 | 0,00 | 0,00 |
| | **8** | 0,00 | 0,31 | 11,58 | 92,77 | 100,01 | 6,00 | 3,40 | 0,20 | 0,20 |
| | **12** | 561,60 | 0,83 | 31,17 | 97,09 | 102,29 | 5,80 | 1,40 | 0,00 | 0,00 |
| | **15** | 1339,72 | 2,71 | 9,90 | 33,82 | 40,62 | 5,20 | 3,60 | 1,40 | 1,40 |
| | **18** | 2732,74 | 10,14 | 11,34 | 23,35 | 25,21 | 4,00 | 2,80 | 2,40 | 2,20 |
| | **20** | 2285,98 | 10,54 | 9,74 | 16,38 | 19,62 | 4,20 | 4,00 | 2,80 | 2,20 |
| **S** | **5** | 0,00 | 0,00 | 19,56 | 287,95 | 290,87 | 6,00 | 4,80 | 1,00 | 0,00 |
| | **8** | 0,00 | 0,00 | 17,54 | 120,98 | 127,45 | 6,00 | 2,60 | 0,00 | 0,00 |
| | **12** | 561,03 | 0,66 | 34,56 | 100,21 | 105,66 | 5,80 | 1,60 | 0,00 | 0,00 |
| | **15** | 1823,85 | 3,72 | 14,55 | 37,25 | 39,13 | 4,80 | 2,80 | 1,00 | 0,80 |
| | **18** | 2049,65 | 10,04 | 9,44 | 35,56 | 35,80 | 4,80 | 4,40 | 1,40 | 1,00 |
| | **20** | 3308,70 | 15,09 | 6,30 | 14,07 | 16,15 | 3,80 | 4,80 | 3,00 | 3,00 |
| **R** | **5** | 0,00 | 2,26 | 0,04 | 55,06 | 56,31 | 5,00 | 6,00 | 3,00 | 3,00 |
| | **8** | 0,00 | 0,78 | 4,92 | 53,67 | 55,08 | 6,00 | 5,20 | 2,00 | 1,00 |
| | **12** | 2159,02 | 1,00 | 21,37 | 48,95 | 49,91 | 5,80 | 2,80 | 0,80 | 0,80 |
| | **15** | 1261,10 | 4,56 | 23,41 | 36,97 | 39,71 | 5,20 | 3,00 | 1,00 | 1,00 |
| | **18** | 1911,66 | 3,99 | 14,86 | 27,27 | 32,70 | 5,00 | 3,00 | 1,00 | 0,80 |
| | **20** | 5992,76 | 16,05 | 8,52 | 11,44 | 11,77 | 3,00 | 3,20 | 4,00 | 3,60 |
| Promedio | | 1443,77 | 4,59 | 14,01 | 67,53 | 70,62 | 5,13 | 3,61 | 1,39 | 1,17 |

Adicionalmente, la Tabla 22 presenta otro resumen de los resultados, pero ordenados por tamaño de la instancias. Acá se puede observar claramente que a medida que aumenta la cantidad de los vértices el modelo empeora, mientras los algoritmos mejoran considerablemente, destacando en mayor medida el VNS por su gap promedio de $14,01\,\%$.

Por tanto, se puede resumir que para tamaños muy pequeños es mejor utilizar el modelo, pero con tamaños de 20 vértices o superiores, es necesario una metaheuristica, como el VNS propuesto.

# Capítulo 6

# Conclusiones

Esta memoria de título propone un modelo de programación lineal entera mixta y un algoritmo del basado en el Variable Neighborhood Search para abordar el problema flexible del vendedor viajero con drones. Se realizó una revisión de la literatura enfocada en los métodos resolutivos de distintos autores. Además, se evaluaron diversas instancias con distintas complejidades, diferenciándose según cantidad de nodos, distancias entre nodos y capacidades de los drones.

A modo general la comparación realizada con el *Set large* de instancias, muestra que el algoritmo propuesto entrega mejores resultados para los resultados medios, a medida que se resuelven las instancias con una menor cantidad de drones. Obteniendo, para el $gap_{avg}$ un promedio de $0,00\,\%$ considerando hasta cuatro drones y un promedio de $0,83\,\%$ para cinco drones. Mientras que si se comparan los resultados mínimos obtenidos por los algoritmos, el VNS es el que mejor se desempeña para las instancias abordadas con uno o dos drones, siendo superado por el SA en las resoluciones con tres o más drones. Esto se refleja en el $gap_{min}$ promedio para el VNS, donde a partir del uso de tres drones el gap se incremente desde $0,21\,\%$ hasta $5,69\,\%$. Es destacable la diferencia de tiempo promedio de ejecución entre el VNS y el GA para resolver el problema, donde el orden de magnitud del $gap_t$ del GA está en la escala de miles. Esto último evidencia la eficiencia del algoritmo propuesto. La prueba de Wilcoxon muestra que los resultados superiores de VNS son estadísticamente significativos en comparación a los resultados de los algoritmos de la literatura.

La comparación realizada con el *Set small* verifica que el modelo propuesto puede

obtener soluciones óptimas para instancias pequeñas en un corto periodo de tiempo, específicamente, para instancias con 5 y 8 vértices. Mientras que para el resto de las instancias el MILP obtiene solo soluciones factibles, evidenciando que al evaluar instancias con mayor cantidad de clientes se debe utilizar una metaheurística.

En trabajos futuros se pueden abordar varios aspectos de ambas propuestas realizadas. Una posible mejora para el algoritmo, es lograr que obtenga mejores resultados para instancias evaluadas con una mayor cantidad de drones. Para realizar esto, una posible estrategia a seguir es intensificar los operadores de búsqueda local y perturbación, de forma que en una iteración se evalué una mayor cantidad de rutas para los vehículos aéreos. Por el lado del modelo de programación lineal entera mixta propuesto, es evidente que la cantidad de vértices que tiene el problema es una limitante. Por tanto, una estrategia para mejorar es agregar restricciones adicionales de reforzamiento para mejorar el rendimiento computacional y tambié, se podría proponer un nuevo algoritmo exacto basado en un *Branch & Cut* o un *Branch & Price*.

Adicionalmente, se pueden realizar esfuerzos para aplicar el modelamiento del problema en condiciones reales de operación entre un camión y drones. En algunos casos implicaría la modificación de los supuestos del FDTSP debido a la complejidad de una operación real. Por ende, también se requeriría la modificación del modelo y el algoritmo propuesto.

# Referencias

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981.

AlMuhaideb, S., Alhussan, T., Alamri, S., Altwaijry, Y., Aljarbou, L., & Alrayes, H. (2021). Optimization of truck-drone parcel delivery using metaheuristics. *Applied Sciences*, 11(14).

Carrasco, J., García, S., Rueda, M., Das, S., & Herrera, F. (2020). Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665.

Cavani, S., Iori, M., & Roberti, R. (2021). Exact methods for the traveling salesman problem with multiple drones. *Transportation Research Part C: Emerging Technologies*, 130:103280.

Chung, S. H., Sah, B., & Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers Operations Research*, 123:105004.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

Guthrie, C., Fosso-Wamba, S., & Arnaud, J. B. (2021). Online consumer resilience during a pandemic: An exploratory study of e-commerce behavior before, during and after a covid-19 lockdown. *Journal of Retailing and Consumer Services*, 61:102570.

Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 86:597–621.

Helsgaun, K. (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J. M., & Brunese, P. A. (2019). Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers  Industrial Engineering*, 129:14–30.

Kuo, R., Edbert, E., Zulvia, F. E., & Lu, S.-H. (2023). Applying NSGA-II to vehicle routing problem with drones considering makespan and carbon emission. *Expert Systems with Applications*, 221:119777.

Lin, S. & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516.

Lu, S.-H., Kuo, R., Ho, Y.-T., & Nguyen, A.-T. (2022). Improving the efficiency of last-mile delivery with the flexible drones traveling salesman problem. *Expert Systems with Applications*, 209:118351.

Luo, Z., Poon, M., Zhang, Z., Liu, Z., & Lim, A. (2021). The multi-visit traveling salesman problem with multi-drones. *Transportation Research Part C: Emerging Technologies*, 128:103172.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers  Operations Research*, 24(11):1097–1100.

Mohd Daud, S. M. S., Mohd Yusof, M. Y. P., Heo, C. C., Khoo, L. S., Chainchel Singh, M. K., Mahmood, M. S., & Nawawi, H. (2022). Applications of drone in disaster management: A scoping review. *Science  Justice*, 62(1):30–42.

Murray, C. C. & Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109.

Murray, C. C. & Raj, R. (2020). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, 110:368–398.

Nwaogu, J. M., Yang, Y., Chan, A. P., & Chi, H.-L. (2023). Application of drones in the architecture, engineering, and construction (AEC) industry. *Automation in Construction*, 150:104827.

Rejeb, A., Abdollahi, A., Rejeb, K., & Treiblmaier, H. (2022). Drones in agriculture: A review and bibliometric analysis. *Computers and Electronics in Agriculture*, 198:107017.

Roberti, R. & Ruthmair, M. (2021). Exact methods for the traveling salesman problem with drone. *Transportation Science*, 55(2):315–335.

Roberts, N. B., Ager, E., Leith, T., Lott, I., Mason-Maready, M., Nix, T., Gottula, A., Hunt, N., & Brent, C. (2023). Current summary of the evidence in drone-based emergency medical services care. *Resuscitation Plus*, 13:100347.

Salama, M. R. & Srinivas, S. (2022). Collaborative truck multi-drone routing and scheduling problem: Package delivery with flexible launch and recovery sites. *Transportation Research Part E: Logistics and Transportation Review*, 164:102788.

Tiniç, G. O., Karasan, O. E., Kara, B. Y., Campbell, J. F., & Ozel, A. (2023). Exact solution approaches for the minimum total cost traveling salesman problem with multiple drones. *Transportation Research Part B: Methodological*, 168:81–123.

Tu, P. A., Dat, N. T., & Dung, P. Q. (2018). Traveling salesman problem with multiple drones. En *Proceedings of the 9th International Symposium on Information and Communication Technology*, SoICT '18, p. 46–53, New York, NY, USA. Association for Computing Machinery.

Zhang, J. & Woensel, T. V. (2023). Dynamic vehicle routing with random requests: A literature review. *International Journal of Production Economics*, 256:108751.

**UNIVERSIDAD DE CONCEPCION - FACULTAD DE INGENIERIA**

**RESUMEN DE MEMORIA DE TITULO**

| | | |
|---|---|---|
| **Departamento** | : | Departamento de Ingeniería Industrial |
| **Carrera** | : | Ingeniería civil Industrial |
| **Nombre del memorista** | : | Benjamín Enrique Brandt Mieres |
| **Título de la memoria** | : | Un Modelo y Algoritmo para el Problema Flexible del Vendedor Viajero con Múltiples Drones |
| **Fecha de la presentación oral** | : | 31/08/2023 |
| **Profesor Guía** | : | Carlos Contreras Bolton |
| **Profesor Revisor** | : | Lorena Pradenas Rojas |
| **Concepto** | : | |
| **Calificación** | : | |


Resumen

Recientemente han existido desarrollos logísticos importantes. Investigaciones respaldan el uso de drones para la entrega de paquetes. Esta memoria de título aborda el problema flexible del vendedor viajero con múltiples drones (FDTSP por sus siglas en inglés: *Flexible Drones Traveling Salesman Problem*). Este problema, caracterizado por su complejidad computacional, busca la combinación óptima que minimice el tiempo para realizar entregas entre un camión y múltiples drones. En este estudio se propone un modelo de programación lineal entera mixta y un algoritmo variable neighborhood search (VNS que opera en dos fases. La primera fase optimiza la ruta del camión mediante operadores basados en el problema del vendedor viajero. La segunda fase utiliza un algoritmo simulated annealing para mejorar las rutas de los drones. El modelo y algoritmo son validados en dos conjuntos de instancias que representan distintas condiciones de operación. Los resultados computacionales muestran que el algoritmo propuesto supera a los algoritmos de la literatura en la mayoría de las instancias evaluadas. Sin embargo, el rendimiento empeora a medida que se utilizan más drones. En tanto, el modelo logra garantizar la optimalidad en instancias de hasta ocho vértices, evidenciando que para instancias con mayor cantidad vértices se debe usar una metaheurísitica.