FACULTAD DE INGENIERÍA - NÚCLEO DE INGENIERÍA BIOMÉDICA,
UNIVERSIDAD DE LA REPÚBLICA

DROMBÓ: Optimización de logística sanitaria de traslados de hasta 3Kg en drones de largo alcance para policlínicas periféricas en el departamento de Tacuarembó

En cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República.

Informe de Proyecto de Grado presentado por

Mariana Bartesaghi

Enrique Castro

Supervisor

Sandro Moscatelli

Núcleo de Ingeniería Biomédica

Franco Simini

Montevideo, Uruguay

9 de septiembre de 2025

Resumen

En el departamento de Tacuarembó, Uruguay, la distribución de insumos médicos enfrenta desafíos logísticos derivados de la dispersión geográfica de policlínicas rurales y las limitaciones en transporte y horarios. Para abordar esta problemática, se planteó el desarrollo de un sistema que automatice la planificación de vuelos con drones, optimizando recursos y mejorando la eficiencia en la logística sanitaria.

La solución propuesta se basó en un algoritmo heurístico de inserción secuencial inspirado en Solomon (1987), adaptado al Problema de Ruteo de Vehículos (VRP) con restricciones de urgencia, capacidad, ventanas de tiempo y recogida y entrega. Además, se diseñó una arquitectura robusta con una API centralizada, base de datos relacional y una aplicación web dirigida al personal de policlínicas y pilotos de dron, integrándose con plataformas externas como RigiTech.

Las pruebas realizadas validaron la funcionalidad del sistema y su capacidad para operar bajo altas cargas concurrentes, logrando tiempos de respuesta adecuados y rutas eficientes en pocos milisegundos de cómputo. Asimismo, se identificó un beneficio administrativo inmediato: la automatización elimina la necesidad de personal dedicado exclusivamente a la coordinación manual de vuelos, reduciendo costos y mejorando la escalabilidad.

El proyecto constituye un primer paso hacia la implementación real de un sistema de gestión de traslados médicos mediante drones en Tacuarembó, abriendo líneas de trabajo futuras en la evaluación empírica de su impacto logístico, la incorporación de heurísticas de mejora y la expansión hacia escenarios con múltiples drones.

Palabras clave: drones, logística sanitaria, Tacuarembó, planificación de traslados, vehicle routing problem (VRP), heurística de inserción, ventanas de tiempo, capacidad, recogida, entrega, ruteo de vehículos, plataforma RigiTech, API REST, optimización de rutas.

Índice general

Glosario

ADS-B *Automatic Dependent Surveillance–Broadcast*: emisión/recepción de posición y velocidad para vigilancia aérea.

AGL *Above Ground Level*: altitud respecto al terreno.

algoritmo de inserción Heurística constructiva que arma rutas insertando iterativamente clientes en la mejor posición factible según una función de costo.

AMSL *Above Mean Sea Level*: altitud respecto al nivel medio del mar.

BVLOS *Beyond Visual Line of Sight*: operación más allá de línea visual.

capacidad Límite máximo de carga que puede transportar el dron, expresado en kilogramos (masa) y litros (volumen). Debe cumplirse durante todo el viaje, no sólo al despegar.

costo Medida a minimizar en la planificación. Combina distancia, tiempo total de operación y penalizaciones por esperas o tardanzas.

CVRP *Capacitated Vehicle Routing Problem*: VRP con restricción de capacidad.

CVRPTW *CVRP with Time Windows*: VRP con ventanatiempo.

demanda Cantidad de unidades físicas requeridas de algún insumo médico para transporte asociado a una policlínica. En VRP se considera para verificar factibilidad de capacidad.

depósito Punto logístico central desde el que parte y al que retorna el dron en cada ruta. En este caso representa al Hospital de Tacuarembó.

dron Aeronave no tripulada operada a distancia o con autonomía supervisada..

envio Operación de carga de insumos en una policlínica que debe preceder a su entrega correspondiente en el Hospital.

GNSS *Global Navigation Satellite System*: sistema de posicionamiento satelital.

pedido Solicitud de abastecimiento de insumos médicos realizada para una policlínica..

plan de vuelo Conjunto de rutas programadas para uno o varios días..

policlínica Punto de origen y/o destino de las solicitudes dentro de la red sanitaria. Se modela como nodo con ventana de tiempo y tiempo de servicio.

ruta Secuencia ordenada de visitas que comienza y termina en el Hospital y puede incluir varias solicitudes de traslados.

ruteo Construcción de rutas que visitan un conjunto de puntos cumpliendo restricciones y minimizando el costo.

tiempo de servicio Duración requerida para atender un pedido o envío una vez el dron arriba a la policlínica ya sea para la carga o descarga del dron o recambio de baterías..

tiempo de vuelo Tiempo efectivo de operación aérea entre despegue y aterrizaje..

traslado Operación que traslada insumos médicos desde el Hospital a una policlínica (Pedido) o viceversa (Envío)..

trayecto Segmento elemental del recorrido entre dos puntos consecutivos..

TSP *Travelling Salesman Problem*: problema clásico de optimización combinatoria que busca la ruta más corta para que un viajante visite un conjunto de ciudades exactamente una vez y retorne a la ciudad de origen.

usuario Persona que utiliza el sistema de planificación para crear solicitud de traslado.

ventana de tiempo Intervalo dentro del cual debe comenzar el servicio en una policlinica. Las llegadas tempranas implican espera; las tardías son inviables o penalizadas.

vertipuerto Infraestructura destinada al despegue, aterrizaje y operación de drones..

viaje Trayecto recorrido entre dos policlínicas o una policlínica y el Hospital de Tacuarembó.

VRP *Vehicle Routing Problem*: familia de problemas de ruteo de vehículos con objetivos de costo y restricciones operativas.

VRPPDTW *Vehicle Routing Problem with Pickup and Delivery and Time Windows*: VRP con recogida, entrega y ventanatiempo.

Siglas

ADS-B *Automatic Dependent Surveillance–Broadcast*: emisión/recepción de posición y velocidad para vigilancia aérea.

AGL *Above Ground Level*: altitud respecto al terreno.

algoritmo de inserción Heurística constructiva que arma rutas insertando iterativamente clientes en la mejor posición factible según una función de costo.

AMSL *Above Mean Sea Level*: altitud respecto al nivel medio del mar.

BVLOS *Beyond Visual Line of Sight*: operación más allá de línea visual.

capacidad Límite máximo de carga que puede transportar el dron, expresado en kilogramos (masa) y litros (volumen). Debe cumplirse durante todo el viaje, no sólo al despegar.

costo Medida a minimizar en la planificación. Combina distancia, tiempo total de operación y penalizaciones por esperas o tardanzas.

CVRP *Capacitated Vehicle Routing Problem*: VRP con restricción de capacidad.

CVRPTW *CVRP with Time Windows*: VRP con ventanatiempo.

demanda Cantidad de unidades físicas requeridas de algún insumo médico para transporte asociado a una policlínica. En VRP se considera para verificar factibilidad de capacidad.

depósito Punto logístico central desde el que parte y al que retorna el dron en cada ruta. En este caso representa al Hospital de Tacuarembó.

dron Aeronave no tripulada operada a distancia o con autonomía supervisada..

envio Operación de carga de insumos en una policlínica que debe preceder a su entrega correspondiente en el Hospital.

GNSS *Global Navigation Satellite System*: sistema de posicionamiento satelital.

pedido Solicitud de abastecimiento de insumos médicos realizada para una policlínica..

plan de vuelo Conjunto de rutas programadas para uno o varios días..

policlínica Punto de origen y/o destino de las solicitudes dentro de la red sanitaria. Se modela como nodo con ventana de tiempo y tiempo de servicio.

ruta Secuencia ordenada de visitas que comienza y termina en el Hospital y puede incluir varias solicitudes de traslados.

ruteo Construcción de rutas que visitan un conjunto de puntos cumpliendo restricciones y minimizando el costo.

tiempo de servicio Duración requerida para atender un pedido o envío una vez el dron arriba a la policlínica ya sea para la carga o descarga del dron o recambio de baterías..

tiempo de vuelo Tiempo efectivo de operación aérea entre despegue y aterrizaje..

traslado Operación que traslada insumos médicos desde el Hospital a una policlínica (Pedido) o viceversa (Envío)..

trayecto Segmento elemental del recorrido entre dos puntos consecutivos..

TSP *Travelling Salesman Problem*: problema clásico de optimización combinatoria que busca la ruta más corta para que un viajante visite un conjunto de ciudades exactamente una vez y retorne a la ciudad de origen.

usuario Persona que utiliza el sistema de planificación para crear solicitud de traslado.

ventana de tiempo Intervalo dentro del cual debe comenzar el servicio en una policlinica. Las llegadas tempranas implican espera; las tardías son inviables o penalizadas.

vertipuerto Infraestructura destinada al despegue, aterrizaje y operación de drones..

viaje Trayecto recorrido entre dos policlínicas o una policlínica y el Hospital de Tacuarembó.

VRP *Vehicle Routing Problem*: familia de problemas de ruteo de vehículos con objetivos de costo y restricciones operativas.

VRPPDTW *Vehicle Routing Problem with Pickup and Delivery and Time Windows*: VRP con recogida, entrega y ventanatiempo.

1. Introducción

En el departamento de Tacuarembó, Uruguay, la distribución de insumos médicos entre el Hospital y las policlínicas rurales enfrenta importantes desafíos logísticos. La dispersión geográfica de los centros de atención, las limitaciones en la capacidad de transporte y los horarios restringidos de atención dificultan garantizar entregas oportunas y eficientes. A esto se suman condiciones variables de demanda y la necesidad de coordinar múltiples traslados diarios, factores que incrementan la complejidad de la gestión.

En los últimos años, el uso de aeronaves no tripuladas (drones) ha emergido como una alternativa viable para mejorar la cobertura y la velocidad en la entrega de insumos médicos, especialmente en regiones con infraestructura vial limitada o de difícil acceso. El Hospital de Tacuarembó ha incorporado un dron para apoyar este tipo de traslados, pero la planificación de los vuelos se realiza actualmente de forma manual, requiriendo tiempo y recursos humanos especializados, y sin asegurar el mejor aprovechamiento de las capacidades del equipo.

La ausencia de un sistema que asista en la organización diaria de los vuelos y que considere simultáneamente las restricciones operativas, la ubicación de los puntos de entrega y la disponibilidad de recursos, genera ineficiencias y puede afectar la calidad del servicio prestado. Surge así la necesidad de contar con una herramienta que permita gestionar y coordinar estas operaciones de manera más ágil, confiable y escalable.

En este contexto, el presente proyecto de grado tiene como objetivo desarrollar una solución tecnológica que automatice la planificación de vuelos de un dron destinado al transporte de insumos médicos entre el Hospital de Tacuarembó y sus policlínicas rurales, optimizando el uso de los recursos disponibles y mejorando la eficiencia de la logística sanitaria en la región. Para ello se implementó un algoritmo heurístico de inserción basado en la propuesta de Solomon[1] a la solución de un problema de Problema de Ruteo de Vehículos (VRP), con restricciones de capacidad, ventanas de tiempo y recogida y entrega.

A lo largo del informe se presenta el dron que se utiliza en el Hospital de Tacuarembó (Capitulo 2), los fundamentos teóricos utilizados (Capítulo 3), la descripción del problema y

la solución propuesta (Capítulo 4), la implementación técnica del sistema (Capítulo 5), las pruebas de validación (Capítulo 6) y, finalmente, las conclusiones y líneas de trabajo futuras (Capítulo 7), cerrando así una propuesta concreta para mejorar la eficiencia en la logística sanitaria mediante el uso de drones.

2. Drones

Definición

Los drones son dispositivos voladores controlados a distancia, utilizados para llevar a cabo diversas tareas. También se les conoce como vehículos aéreos no tripulados o aeronaves controladas remotamente. Según la Organización de Aviación Civil Internacional[2], un dron se define como un conjunto de componentes configurables que incluyen: un avión controlado a distancia, la estación de control correspondiente, los enlaces de comunicación necesarios para su manejo, y cualquier otro elemento que sea necesario durante el vuelo.

En términos amplios, los drones son vehículos voladores que pueden clasificarse en distintas categorías, presentando una amplia variedad de especificaciones, capacidades y características.

Los drones son herramientas multifuncionales que se utilizan en una variedad de sectores y aplicaciones. Desde su uso recreativo por aficionados hasta su empleo en misiones militares, los drones han revolucionado numerosos campos. En el ámbito civil, se emplean para labores comerciales como la entrega de paquetería, la cobertura mediática de eventos y la vigilancia de infraestructuras. Además, tienen un papel fundamental en actividades gubernamentales, como la supervisión del medio ambiente, la agricultura de precisión y la gestión de desastres naturales.

En cuanto a su clasificación, los drones se diferencian por su uso (militar o civil) y por el método de control utilizado. En términos de control, pueden ser autónomos, operados remotamente por un piloto humano, o seguir rutas preprogramadas. Esta diversidad de funciones y formas de operación los hace adecuados para una amplia gama de aplicaciones, desde misiones de reconocimiento hasta la realización de tareas de inspección y monitoreo en áreas de difícil acceso[2].

2.1. Eager-03

Los drones médicos pueden volar largas distancias sobre terrenos peligrosos para entregar medicamentos, vacunas, sangre y kits de diagnóstico. Los drones pueden ayudar a fortalecer los sistemas nacionales de atención médica y conectar hospitales y policlínicas con pacientes, mejorando los resultados y reduciendo los tiempos de espera, especialmente en comunidades remotas. Para que las cadenas de suministro de atención médica cambien a drones, deben ser rentables y fáciles de usar. El uso de drones para el traslado de suministros médicos está mejorando con el tiempo. Al principio, pequeños estudios piloto[3] han demostrado que cuando el gobierno, las organizaciones de ayuda humanitaria y los actores involucrados en el sistema de drones trabajan juntos, pueden mejorar el acceso a la atención médica en comunidades remotas, rurales y locales en todo el mundo.

En el ámbito de la salud, el empleo de drones para la distribución se ha vuelto factible, como lo evidencian diversas iniciativas que destacan las ventajas de utilizar drones en el transporte de sangre, equipamiento médico y medicamentos[4].

El dron modelo Eiger-03, fabricado y distribuido por la empresa RigiTech, representa un avance significativo en el campo de la tecnología de drones aplicada a la atención médica. En la Figura 2.1 se muestra este modelo, destaca por su capacidad para abordar desafíos logísticos en áreas de difícil acceso o con infraestructura limitada.

Equipado con tecnología avanzada de navegación y sistemas de control de vuelo, el dron puede operar de manera autónoma y precisa, incluso en condiciones adversas.

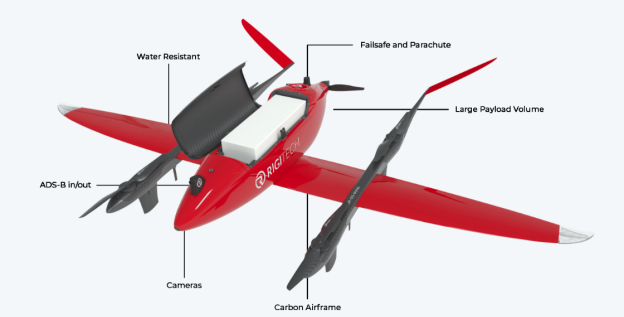

Figura 2.1: Estructura dron Eiger-03. Tomado de Rigitech[5].

Características físicas

El dron presenta una serie de características específicas que garantizan su eficiencia y seguridad en este contexto particular. El Eiger-03 es un dron de porte menor, con un peso que varía entre los 14kg (vacío) y 18kg (con baterías a bordo), cuenta con una capacidad de carga de 3kg por lo que tiene un peso de lanzamiento máximo de 21kg en total.

El alcance del dron, de hasta 100km, lo hace idóneo para llevar medicamentos a zonas remotas o de difícil acceso, donde el acceso terrestre puede ser limitado o inexistente. Las dimensiones del compartimento de carga, de 45 cm x 16 cm x 21 cm, permiten el transporte de diferentes tipos de medicamentos, desde suministros básicos hasta productos más especializados.

El dron tiene una capacidad para operar hasta una altitud máxima de 3000m sobre el nivel del mar (AMSL). Sin embargo, es importante tener en cuenta que, debido a razones de seguridad, no se permite volar durante tormentas, lluvias o heladas, garantizando la integridad de los medicamentos y la seguridad de las operaciones.

Además, equipado con dos cámaras, el Eiger permite a los operadores remotos realizar un seguimiento en tiempo real de sus operaciones. La cámara frontal ofrece una visión clara del vuelo, capturando imágenes nítidas del entorno y asegurando una vista despejada del camino por delante. La cámara orientada hacia abajo facilita el monitoreo durante el despegue y el aterrizaje, proporcionando una visibilidad crucial para operaciones fluidas y seguras.

Capacidad

El modelo Eiger-03 cuenta con un compartimento de carga con una capacidad máxima de 15 litros de volumen y 3 kilogramos de peso, lo que significa que cualquier insumo transportado debe cumplir ambas restricciones de forma simultánea. Esta diseñado para transportar insumos que no requieran condiciones especiales de temperatura, como medicamentos e instrumentos médicos. Para aquellos insumos de carácter sensible, como sangre o leche, el dron cuenta con compartimentos especializados que cumplen con la normativa UN3373 [6] para el transporte seguro de muestras médicas. Estos compartimentos ayudan a controlar la tempe-

ratura del insumo trasladado y están diseñados con una caja de transporte que protege los insumos sensibles durante todo el trayecto. En la Figura 2.2 podemos ver el contenedor de carga del dron, un compartimento y sus características.

Maximum Payload Weight: 3 kg

(L x W x H)	Cardboard box	Medical Payload boxes (UN3373)
External dimensions	45 cm x 16 cm x 21 cm	(3x) 15 cm x 16 cm x 21 cm
Volume	15 L	150 - 300 blood samples
Reference Image		

Figura 2.2: Contenedores de carga de 3 Kg del dron Eiger-03[5].

Rendimiento de vuelo

El rendimiento del vuelo de un dron está sujeto a una serie de variables, entre las que se incluye la carga útil que transporta. Esta carga puede influir significativamente en la distancia máxima que el dron puede recorrer con eficacia. Para ilustrar esta relación, presentamos las tablas 2.1 y 2.2 que diferencian entre viajes unidireccionales y viajes de ida y vuelta. Se han establecido condiciones iniciales estándar, como baterías totalmente cargadas y condiciones climáticas normales, para una comparación precisa.

Carga	0 kg	1 kg	2 kg	3 kg
Rango máximo de vuelo	100 km	90 km	85 km	80 km
Tiempo de vuelo	57 min	52 min	49 min	46 min
Velocidad	16 m/s	17.3 m/s	17.3 m/s	17.4 m/s

Tabla 2.1: Vuelo de ida. [7]

Carga	0 kg	1 kg	2 kg	3 kg
Rango máximo de vuelo	45 km	40 km	37 km	35 km

Tabla 2.2: Vuelo de ida y vuelta[7].

Límites Operacionales

El dron Eiger-03 tiene especificaciones operativas detalladas en la tabla 2.3 que incluyen velocidades de vuelo para configuraciones tanto de multicóptero, que es una aeronave con varios rotores horizontales capaces de despegar y aterrizar verticalmente y mantenerse en vuelo estacionario, aunque con menor autonomía y velocidad, como de ala fija, que emplea alas similares a las de un avión convencional para generar sustentación mediante el avance, ofreciendo mayor eficiencia y velocidad en vuelo horizontal pero sin la capacidad de mantenerse estático en el aire.

Categoría	Para Multicóptero	Para Ala Fija
Velocidad de Crucero	6 m/s	29 m/s
Velocidad Máxima de Vuelo	10 m/s	35 m/s
Velocidad Mínima de Vuelo	0 m/s	23 m/s
Velocidad de Ascenso	3.5 m/s	3.3 m/s
Velocidad de Descenso	2.7 m/s	4.5 m/s
Resistencia Máxima al Viento	12 m/s	15 m/s

Tabla 2.3: Relación Velocidad-Modalidad[7].

Velocidad de Crucero: La velocidad a la que el dron puede mantener un vuelo constante y estable durante períodos prolongados de tiempo. Es una velocidad eficiente en términos de consumo de energía y permite al dron cubrir distancias de manera efectiva.

Velocidad Máxima de Vuelo: La velocidad máxima que el dron puede alcanzar en vuelo.

Velocidad Mínima de Vuelo: La velocidad mínima a la que el dron puede mantenerse en vuelo sin perder elevación ni control. Es importante para operaciones que requieren maniobras precisas o vuelo estacionario.

Velocidad de Ascenso: La velocidad a la que el dron puede subir en el aire. Es importante para superar obstáculos o alcanzar altitudes específicas rápidamente.

Velocidad de Descenso: La velocidad a la que el dron puede descender en el aire. Es importante para un descenso controlado y seguro, especialmente durante aterrizajes.

Resistencia Máxima al Viento: La velocidad máxima del viento con la que el dron puede operar de manera segura y estable. Es importante para garantizar la estabilidad y el control del dron en condiciones de viento fuerte.

Aterrizaje

El proceso de aterrizaje del dron Eiger-03 utiliza distintos sensores y sistemas de navegación para garantizar que el dron pueda descender de forma segura y controlada. Gracias a la combinación de posicionamiento satelital, sensores de movimiento y cámaras, el dron es capaz de aterrizar correctamente en el lugar previsto.

El dron Eiger-03 utiliza plataformas de aterrizaje, como la que vemos en la Figura 2.3, con marcadores visuales para identificar el punto de descenso. Estas plataformas están instaladas en los lugares definidos para despegue y aterrizaje, y permiten que el dron reconozca su ubicación mediante visión computarizada. Este sistema facilita operaciones más seguras y controladas, incluso en espacios reducidos.

Figura 2.3: Plataformas de Precisión para el aterrizaje de drones[7].

Componentes del Hardware

Los drones se componen de una variedad de sensores que aseguran su funcionamiento eficiente. El dron Eiger-03, en particular, integra los siguientes dispositivos con funciones específicas.

Tráfico Espacial

El sistema Detect and Avoid (DAA) permite a los drones identificar y evitar obstáculos en vuelo, incluyendo otras aeronaves. Utiliza sensores, transpondedores y antenas que captan

señales de vigilancia aérea como ADS-B IN, ayudando a prevenir colisiones en espacio aéreo compartido.

Además, tecnologías como FLARM y los dispositivos de Identificación Remota (RID) refuerzan la seguridad. FLARM detecta aeronaves cercanas, mientras que RID permite identificar y rastrear drones a distancia.

En operaciones BVLOS (más allá de línea visual), la plataforma RigiCloud centraliza datos en tiempo real para monitorear los vuelos y asistir en la toma de decisiones, mejorando la seguridad y eficiencia de las misiones.

Sistema de Navegación Global de Doble Redundancia (Dual Redundant GNSS)

El GNSS[8] (Sistema Global de Navegación por Satélite) permite obtener información precisa de ubicación mediante señales satelitales. El GPS es una de sus variantes más conocidas y utilizadas.

El dron Eiger-03 cuenta con un sistema GNSS con doble módulo, lo que le permite mantener la navegación aún si uno de ellos falla o pierde señal.

ADS-B

El sistema ADS-B (Automatic Dependent Surveillance–Broadcast) permite que los drones transmitan su posición, altitud y velocidad a otras aeronaves y estaciones terrestres equipadas con receptores compatibles. Esta información se emite mediante señales de radio y puede ser visualizada en tiempo real, lo que mejora la conciencia situacional y contribuye a evitar colisiones en el espacio aéreo compartido.

Además de su función en la seguridad, el ADS-B permite optimizar el tráfico aéreo, ya que los datos precisos que genera facilitan una mejor gestión del flujo por parte de los controladores[9].

Magnetómetros

Los magnetómetros en aviación miden el campo magnético terrestre para ayudar a determinar la orientación de la aeronave. Se dividen en dos tipos: absolutos, que se calibran con constantes internas, y relativos, que requieren un campo magnético de referencia, usualmente mediante un Modelo Magnético Mundial (WMM) [10].

En drones, el magnetómetro trabaja junto a acelerómetros y giroscopios para determinar posición y orientación. Está compuesto por un sensor que detecta el campo magnético y lo convierte en una señal eléctrica, una unidad de procesamiento que interpreta los datos y una interfaz que comunica esta información al controlador de vuelo [11].

IMU

Una Unidad de Medida Inercial (IMU) es un dispositivo que mide aceleraciones lineales y velocidades angulares. Algunos modelos también incluyen magnetómetros.

Las IMUs son componentes clave en drones y otros sistemas aéreos no tripulados, donde se usan para control, estabilización, orientación y navegación.

Los datos crudos de una IMU (aceleraciones, tasas angulares y campo magnético) pueden ser procesados por un Sistema de Navegación Inercial (INS) para estimar posición, orientación y velocidad, mejorando la capacidad de navegación y control [12].

Pitot tube

El tubo de Pitot [13] se utiliza para medir la velocidad de un vehículo en movimiento a partir de la presión del aire o del agua. Funciona comparando la presión estática con la presión total que se genera al enfrentar el flujo.

En aviación, el tubo Pitot-estático o de Prandtl se emplea como velocímetro. Está compuesto por un tubo orientado al flujo de aire y orificios laterales que captan la presión estática[14].

Suele ubicarse en zonas donde reciba el flujo libre, como la nariz o el ala de la aeronave, para asegurar una medición confiable.

Altímetro

El altímetro en un dron mide la altitud respecto al nivel del mar. Su funcionamiento se basa en la presión atmosférica. A medida que el dron asciende o desciende, la presión cambia. El altímetro detecta estos cambios mediante una cápsula sellada que se deforma al variar la presión. Esa deformación se traduce en una lectura de altitud[15].

FLARM

FLARM es un sistema de vigilancia y alerta de tráfico aéreo diseñado para aeronaves

pequeñas y planeadores. Su función principal es prevenir colisiones, intercambiando datos como posición, dirección y velocidad entre aeronaves cercanas. Esto permite detectar posibles conflictos y tomar medidas evasivas a tiempo. A diferencia de los sistemas de radar convencionales, FLARM es más compacto y económico[16].

C2 Link

El C2 Link es el canal de comunicación entre el dron y el operador en tierra, que permite enviar comandos y recibir información de telemetría[17]. La telemetría celular dual permite al dron utilizar dos conexiones móviles (3G o 4G) para enviar datos en tiempo real sobre su estado. Esta redundancia asegura la continuidad de la comunicación incluso si una de las redes falla.

2.2. Marco regulatorio para la operativa en Uruguay

El marco regulatorio para el uso de drones en Uruguay, específicamente en la ciudad de Tacuarembó, se encuentra establecido en la Resolución de la Dirección Nacional de Aviación Civil e Infraestructura Aeronáutica (DINACIA)[18] número 2981. Esta resolución, emitida por el Director Nacional de Aviación Civil e Infraestructura Aeronáutica, establece las normativas y condiciones para la operación de Dispositivos Aéreos Operados a Distancia (RPAS) en territorio uruguayo.

La Resolución 2981 clasifica los dispositivos aéreos operados a distancia en tres categorías:

- **Menores:** Son aquellos que tienen un peso de lanzamiento de hasta 25 kg, están destinados exclusivamente al deporte o la recreación. No requieren registro, certificado de aeronavegabilidad ni licencia, autorización o permiso para el operador. Sin embargo, están sujetos a ciertas restricciones, como no operar en espacios aéreos controlados o en zonas de tráfico de aeródromos, ni sobrepasar los 120 metros sobre el nivel del suelo (AGL). La operación debe realizarse en condiciones de visibilidad meteorológica (VMC) y en línea directa de vista.
- **Medianos:** Con un peso de lanzamiento de más de 25 kg y hasta 260 kg de peso vacío inclusive, también están dedicados exclusivamente al deporte o la recreación. Deben

inscribirse en un registro técnico llevado por la Dirección de Seguridad Operacional, y se les asigna un número correlativo en lugar de matrícula. No se les extiende certificado de aeronavegabilidad ni se requiere licencia aeronáutica para el operador. Sin embargo, el operador debe obtener un "Permiso de Operador de Dispositivo Aéreo Operado a Distancia"después de demostrar conocimientos básicos de normativa aeronáutica, pericia de vuelo y habilidades para resolver situaciones de emergencia. Están sujetos a restricciones similares a los dispositivos menores en cuanto a operación en espacios controlados, altura máxima de vuelo y condiciones meteorológicas.

- **Mayores (o Sistema de Aeronave Pilotada a Distancia - RPAS):** Estos dispositivos, con un peso vacío de más de 260 kg, también están dedicados exclusivamente al deporte o la recreación. Se consideran aeronaves y deben cumplir con la normativa nacional aplicable, además de lo establecido en la Resolución 2981. Para su operación se requiere licencia aeronáutica, además del "Permiso de Operador de Dispositivo Aéreo Operado a Distancia". Los operadores deben demostrar pericia de vuelo y habilidades para resolver situaciones de emergencia. Están sujetos a restricciones similares a los dispositivos menores y medianos en cuanto a operación en espacios controlados, altura máxima de vuelo y condiciones meteorológicas.

La implementación de esta regulación se fundamenta en la necesidad de garantizar niveles aceptables de Seguridad Operacional en el espacio aéreo uruguayo, así como el cumplimiento de la normativa vigente en aspectos administrativos, técnicos y comerciales relacionados con el uso de drones. En el caso del dron Eiger-03 recae sobre el marco regulatorio para dron de porte menor, ya que su peso máximo de lanzamiento es de 21kg.

3. Estado del arte

3.1. Relevamiento de estudios similares

En los últimos años, la creciente utilización de drones para tareas logísticas en contextos sanitarios ha impulsado el desarrollo de modelos de optimización orientados a resolver problemas de planificación de rutas. En particular, múltiples estudios han abordado esta temática aplicando distintas variantes del problema de ruteo de vehículos (VRP, por sus siglas en inglés), adaptadas a las particularidades del transporte aéreo no tripulado. Esta línea de investigación ha demostrado ser especialmente relevante en situaciones de emergencia, como pandemias o desastres naturales, donde la rapidez en la entrega, la eficiencia en el uso de recursos y la capacidad de acceder a zonas remotas resultan determinantes.

Por ejemplo, Du et al. (2022)[19] analizan el uso de drones para entregar suministros médicos durante la pandemia de COVID-19, y plantean el problema como uno de ruteo con restricciones específicas del contexto sanitario. En su trabajo, se destaca la necesidad de planificar cuidadosamente las rutas para maximizar el aprovechamiento de los dispositivos disponibles, lo cual se traduce en la aplicación de un enfoque de optimización que toma como base el modelo VRP, incorporando elementos como la capacidad limitada y las distancias máximas admisibles.

Por su parte, Shi et al. (2022)[20] proponen una variante del VRP que considera no solo la entrega de insumos, sino también la recolección simultánea de muestras u otros materiales médicos. El modelo que desarrollan permite atender múltiples puntos con un mismo dron, buscando un equilibrio entre la eficiencia operativa y el uso mínimo de recursos. Esta perspectiva refuerza la idea de que las soluciones logísticas con drones requieren adaptaciones específicas de modelos tradicionales de ruteo para contemplar las nuevas condiciones tecnológicas y operativas.

En una línea complementaria, Huang, Zhang y Su (2023)[21] documentan experiencias reales de distribución médica mediante drones en zonas de difícil acceso, como regiones montañosas o áreas insulares. Si bien el enfoque es principalmente empírico, su análisis

refleja la importancia de la planificación de rutas eficientes para asegurar la cobertura y la continuidad de la atención sanitaria. De forma implícita, su trabajo también se inscribe dentro del enfoque general de los problemas de ruteo, al abordar decisiones clave sobre asignación de destinos, recorridos y cobertura geográfica.

En conjunto, estos estudios muestran que la distribución médica mediante drones ha sido ampliamente modelada y analizada a través del marco conceptual del VRP, adaptando sus supuestos y restricciones a las condiciones particulares del transporte aéreo. Las limitaciones de carga, la autonomía de vuelo, la posibilidad de combinar entrega y recolección, así como la necesidad de minimizar el número de dispositivos utilizados, son elementos que se incorporan en las distintas variantes propuestas. Por lo tanto, considerando estos antecedentes, puede afirmarse que el problema abordado en este trabajo se reduce a una formulación particular del problema de ruteo de vehículos (VRP) con las siguientes características: capacidad, ventanas de tiempo, recogida y entrega y urgencia.

3.2. Vehicle Routing Problem

El problema de ruteo de vehículos (VRP, por sus siglas en inglés) es un nombre genérico que se da a toda una clase de problemas relacionados con el diseño óptimo de rutas que debe seguir una flota de vehículos para atender a un conjunto de clientes.

El concepto general de VRP fue introducido por primera vez por Dantzig y Ramser[22] en 1958, donde se plantea el ruteo óptimo de una flota de camiones de entrega de gasolina entre una terminal de almacenamiento y un gran número de estaciones de servicio abastecidas por la terminal. El objetivo del VRP consiste en dado un depósito y una flota de vehículos, determinar un conjunto de rutas de costo mínimo que comiencen y terminen en el depósito de tal forma que todos los clientes sean visitado por un vehículo. Se buscará minimizar tanto el costo de cada ruta como la cantidad de vehículos que se utilizan.

Existen distintas variantes del problema debido a las diferentes restricciones operativas en las rutas, así como a las variaciones en las características de los vehículos, depósitos y clientes.

El depósito es el lugar desde donde comienzan las rutas y generalmente almacena la mercadería que los vehículos deben distribuir. Cada vehículo arranca y termina su recorrido en el depósito. Al igual que los clientes, el depósito también puede tener restricciones de horario que limiten los momentos en los que se pueden realizar las entregas[23].

Los vehículos pueden estar sujetos a restricciones tanto en la duración de sus recorridos como en la cantidad de carga que pueden llevar. Si todos los vehículos de la flota tienen la misma capacidad y condiciones, se considera una flota homogénea; en cambio, si tienen diferencias en estas características, se denomina heterogénea. En la versión más simple del problema, cada vehículo completa una única ruta, aunque existen enfoques en los que pueden realizar múltiples recorridos dentro de un mismo período[23].

G. Laporte [24] plantea formalmente el VRP como el problema de diseñar rutas de entrega de menor costo desde un depósito hasta un conjunto de clientes dispersos geográficamente, sujeto a restricciones adicionales. Este problema es fundamental para la gestión de distribución y debe resolverse rutinariamente por los transportistas. En la práctica, existen varias variantes del problema debido a la diversidad de reglas operativas y restricciones encontradas en aplicaciones de la vida real.

Laporte presenta el VRP mediante la siguiente definición matemática [24]. Sea $G = V \cup A$ un grafo dirigido donde $V = \{0, 1, ..., n\}$ es el conjunto de vértices y $A = \{(i, j) \mid i,\ j \in V,\ i \neq j\}$ es el conjunto de arcos. El vértice 0 representa el depósito mientras que los demás vértices corresponden a los clientes. Una flota de m vehículos idénticos con capacidad Q parten del depósito. El tamaño de la flota se da a priori o es una variable de decisión. Cada cliente i tiene una demanda no negativa q_i. Se define una matriz de costos c_{ij} en A. El VRP consiste en diseñar rutas de m vehículos de manera que cada ruta comience y termine en el depósito, cada cliente sea visitado exactamente una vez por un solo vehículo, y la demanda total de una ruta no exceda Q.

En esta sección nos centraremos exclusivamente en cuatro variantes que presentan las aplicaciones industriales más relevantes, comenzando con Traveling Salesman Problem (TSP) y siguiendo con Capacitated Vehicle Routing Problem (CVRP), Capacitated Vehicle Routing

Problem with Time Windows (CVRPTW) y Pickup and Delivery Problem with Time Windows (VRPPDTW). Cada subsección introduce el problema y posteriormente presenta un modelo matemático del mismo.

TSP: Este clásico problema implica que un viajante debe visitar un conjunto de ciudades exactamente una vez y regresar al punto de partida, con el objetivo de minimizar la distancia total recorrida. A pesar de su simplicidad conceptual, el TSP es fundamental para comprender los principios del ruteo óptimo y tiene aplicaciones prácticas tanto directas como indirectas en la planificación de rutas de vehículos[25].

CVRP: En esta variante, se asume que todos los vehículos tienen una capacidad máxima determinada que puede diferir para cada vehículo de la flota. El desafío es diseñar rutas para los vehículos de manera que se satisfagan todas las demandas de los clientes sin superar la capacidad de carga de los vehículos[26].

CVRPTW: En el CVRPTW[27], el servicio a cualquier cliente debe comenzar dentro de un intervalo de tiempo específico, denominado ventana de tiempo. Este problema añade una dimensión temporal al ruteo de vehículos, incrementando la complejidad de planificar las rutas para adherirse a los horarios establecidos.

VRPPDTW: El problema de ruteo con recolección y entrega de productos con ventanas de tiempo[28] considera escenarios en los que los vehículos no solo transportan mercadería desde un depósito a los clientes, sino que también deben recoger productos en ciertos puntos. La complejidad de este modelo radica en que la carga de los vehículos varía a lo largo del recorrido, en lugar de solo aumentar o disminuir en un único sentido.

El VRP es un problema NP-Hard, donde los enfoques exactos pierden eficiencia a medida que el tamaño de la instancia aumenta. Debido a esta dificultad y a su impacto en la optimización de costos de distribución, se han desarrollado diversas heurísticas a lo largo del tiempo, permitiendo encontrar soluciones de buena calidad en un tiempo razonable.

3.2.1. Travelling Salesman Problem

En su origen el problema del TSP[25] surge buscando dar solución al problema que consiste en que para un conjunto de n ciudades y sus distancias modeladas en una matriz simétrica n por n, encontrar un recorrido de longitud mínima que visite cada ciudad exactamente una vez. Por supuesto, en lugar de distancia, otras nociones como tiempo, costo, etc., pueden ser consideradas; utilizaremos *"distancia"* para representar cualquier medida de ese tipo. Este problema ha sido estudiado durante muchos años, con éxito limitado. Los algoritmos exactos pueden requerir tiempos de ejecución desmesurados; los métodos heurísticos producen buenas respuestas para problemas algo más grandes en tiempos razonables, pero no garantizan que se obtenga la respuesta óptima. Además, con las heurísticas actuales, la efectividad disminuye y el tiempo de ejecución aumenta rápidamente con n.

En la Figura 3.1 se muestra un ejemplo de una instancia del problema (TSP) a la izquierda y a la derecha se muestra la solución óptima cuando se utilizan distancias euclidianas para medir la distancia entre dos ciudades. El problema viene en diferentes variantes dependiendo de qué propiedades satisfagan las distancias. Si las distancias cumplen que la distancia de la ciudad i a la ciudad j es la misma que la distancia de la ciudad j a la ciudad i para todas las ciudades i y j, entonces se dice que el problema es simétrico. Si esta propiedad no se cumple, entonces se dice que el problema es asimétrico.

Se dice que un problema es euclidiano si las ciudades están ubicadas en $\mathbb{R}^d$ y la distancia entre dos ciudades es la distancia euclidiana. El problema se puede formular como un modelo matemático de la siguiente manera:

Sea $G = (V, A)$ un grafo completo y dirigido donde $V = \{1, \ldots, n\}$ es el conjunto de nodos/ciudades y A es el conjunto de arcos. A cada arco $(i, j) \in A$ se le asigna una distancia o costo c_{ij}. Definimos la variable de decisión binaria x_{ij} que se establece en uno si y solo si el arco (i, j) se utiliza en la solución. El problema puede formularse como

$$\text{mín} \sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{ij} x_{ij} \tag{1}$$

con las siguientes restricciones

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 1 \quad \forall i \in V \tag{2}$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} = 1 \quad \forall j \in V \tag{3}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \quad \forall S \subseteq V \tag{4}$$

La función objetivo (1) es minimizar los costos asociados a los arcos del grafo, mientras que las ecuaciones (2) y (3) garantizan que cada nodo tenga exactamente un arco de salida y uno de entrada. Por otro lado, la ecuación (4) se encarga de evitar la formación de subrecorridos en la solución[29].

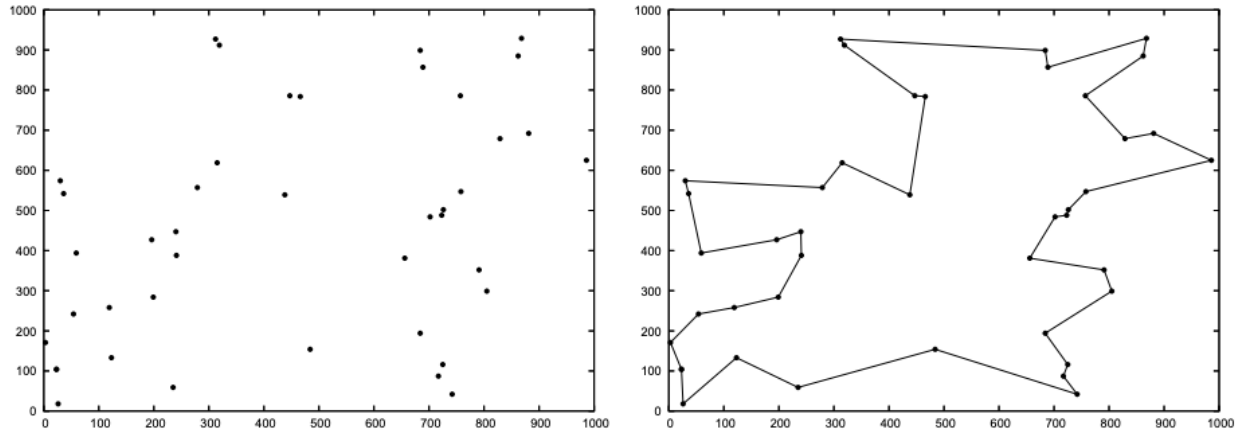

Figura 3.1: Ejemplo de una solución al Problema del Viajante[30].

3.2.2. Capacitated Vehicle Routing Problem

El problema de ruteo de vehículos con capacidad (CVRP)[26], por sus siglas en inglés) es un pilar fundamental en el campo de la optimización combinatoria y desempeña un papel crucial en numerosas aplicaciones logísticas y de transporte. Su complejidad radica en

encontrar un conjunto óptimo de rutas para una flota de vehículos, todos con capacidades limitadas, partiendo desde un único depósito para servir a una serie de clientes dispersos.

El CVRP[31] se conoce por ser NP-completo y generaliza el conocido Problema del Viajante de Comercio (TSP), lo cual requiere la determinación de un ciclo simple de costo mínimo que visite todos los vértices de G (ciclo Hamiltoniano), y surge cuando $Q \geq \Sigma_{i \in V_c} q_i$, siendo Q la capacidad máxima del vehículo.

La formulación matemática del CVRP implica la creación de un modelo que represente adecuadamente la complejidad del problema. Dado una flota de vehículos $R = \{1, \ldots, m\}$, se utiliza un grafo dirigido completo $G = (V, A)$, donde $V = \{0, 1, ..., n, n+1\}$ representa con v_0 la salida desde el depósito y v_{n+1} la llegada al depósito de tal forma que no existan aristas entrantes a v_0 ni aristas salientes de v_{n+1}. Siendo los nodos restantes los clientes del sistema. Por otro lado, al igual que en el problema de TSP, A es el conjunto de arcos que representan las rutas posibles entre sus nodos. A cada arco $(i, j) \in A$ se le asigna un costo o distancia c_{ij}, y se definen variables de decisión binarias x_{ij}, las cuales indican si un arco se utiliza o no en la solución.

$$x_{rij} = \begin{cases} 1, & \text{si el vehiculo } r \text{ utiliza el arco } (i,j) \text{ en la solucion} \\ 0, & \text{en otro caso} \end{cases} \tag{5}$$

Por ultimo, cada vehículo tiene una capacidad Q y cada cliente i tiene una demanda $d_i \leq Q$.

El objetivo del CVRP es minimizar el costo total de las rutas de todos los vehículos, sujetos a ciertas restricciones. Estas restricciones incluyen que cada vehículo debe comenzar y terminar en el depósito, cada cliente debe ser visitado exactamente una vez y la suma de las demandas de los clientes en cada ruta no puede exceder la capacidad del vehículo[32].

La función objetivo que minimiza el costo del viaje es la siguiente

$$\text{mín} \sum_{r=1}^{p} \sum_{i=0}^{n+1} \sum_{j=0, i \neq j}^{n+1} c_{ij} x_{rij} \tag{6}$$

Sujeto a las siguientes restricciones

$$\sum_{r=1}^{m} \sum_{i=0, i\neq j}^{n+1} x_{rij} = 1 \quad \forall j \in \{1, \ldots, n\} \tag{7}$$

$$\sum_{j=0}^{n+1} x_{r0j} = 1 \quad \forall r \in \{1, \ldots, m\} \tag{8}$$

$$\sum_{i=0, i\neq j}^{n+1} x_{rij} = \sum_{i=0}^{n+1} x_{rji} \quad \forall j \in \{1, \ldots, n\}, r \in \{1, \ldots, m\} \tag{9}$$

$$\sum_{i=0}^{n+1} x_{ri(n+1)} = 1 \quad \forall r \in \{1, \ldots, m\} \tag{10}$$

$$\sum_{i=0}^{n} \sum_{j=1, i\neq j}^{n} d_j x_{rij} \leq Q_r \quad \forall r \in \{1, \ldots, m\} \tag{11}$$

$$\sum_{r=1}^{m} \sum_{i\in S} \sum_{j\in S, i\neq j} x_{rij} \leq |S| - 1 \quad \forall S \subseteq \{1, \ldots, n\} \tag{12}$$

Por (7) podemos asegurar que cada cliente es visitado por solamente un vehículo. La restricción (8) nos dice que todos los vehículos parten del depósito. La ecuación (9) garantiza que para cada cliente, la cantidad de aristas entrante es igual a la cantidad de aristas salientes, osea que cada vehículo que llega a un cliente sale del mismo y (10) indica que el vehículo retorna una única vez al depósito.

Por (11) surge una resticción asociada a la demanda de los clientes y la capacidad del vehículo de tal forma que la suma de las demandas de los clientes d_j visitados por el vehículo en la solución óptima debe ser menor o igual a la capacidad de transporte del vehículo Q_r.

Por último debido a la restricción de que cada cliente debe ser visitado exactamente una vez usando (12) planteamos que la solución óptima no debe tener un ciclo en el recorrido de cada uno de los vehículos.

3.2.3. The Capacitated Vehicle Routing Problem with Time Windows

El problema de ruteo de vehículos con capacidad y ventanas de tiempo (CVRPTW) amplía el alcance del CVRP al considerar tiempos de viaje c_{ij} entre los nodos i y j, además de tiempos de servicio s_i y ventanas de tiempo $[e_i, l_i]$ asignadas a cada cliente i.

Cada vértice $v_i \in V$ está asociada una carga no negativa q_i (con $q_0 = 0$), una duración de servicio no negativa d_i (con $d_0 = 0$), y una ventana de tiempo $[e_i, l_i]$, donde e_i y l_i son enteros no negativos y $e_i \leq l_i$. Cada arco (v_i, V_j) tiene asociado un costo no negativo o tiempo de viaje c_{ij}. El CVRPTW consiste en diseñar un conjunto de rutas G de tal manera que:

(I) cada ruta comienza y termina en el depósito

(II) cada cliente pertenece exactamente a una ruta;

(III) la suma de las cargas qi de cada nodo en G que pertenece a una ruta no excede la carga máxima del vehículo asignado (Q_k).

(IV) el servicio al cliente i comienza en el intervalo $[e_i, l_i]$, y cada vehículo sale del depósito y regresa al depósito en el intervalo $[e_0, l_0]$

(V) el tiempo total de viaje de todos los vehículos se minimiza.

El vehículo debe llegar dentro de la ventana de tiempo de un cliente. El problema se puede modelar utilizando el marco de VRP introducido en esta sección. Para facilitar la notación, nuevamente consideramos que el depósito se divide en dos nodos ($i = 0$, $i = n+1$). La ruta $\bar{w} = (v_0, v_1, ..., v_h, v_{h+1})$ debe cumplir con los siguientes criterios para ser válida. El requisito de capacidad es el mencionado anteriormente al introducir el CVRP (11):

$$\sum_{i=0}^{n} \sum_{j=1, i \neq j}^{n} d_j x_{rij} \leq Q_r \quad \forall r \in \{1, \ldots, m\} \tag{13}$$

Introducimos una variable S_i para indicar cuándo comienza el servicio en el nodo i. Una ruta debe cumplir con las siguientes restricciones para ser viable en términos de tiempo:

$$x_{ijr}(s_{ir} + c_{ij} - s_{jr}) \leq 0 \quad \forall i, j \in N, \forall r \in R \tag{14}$$

$$e_i \leq s_{ir} \leq l_i \quad \forall i, j \in N, \forall r \in R \tag{15}$$

La desigualdad (14) indica que el tiempo de comienzo de servicio en un cliente sumado a el tiempo de viaje al cliente vecino debe ser menor o igual a el tiempo de comienzo de

servicio en el cliente vecino y (15) restringe los tiempos de inicio de servicio en cada cliente a estar dentro de su ventana de tiempo asociada[33].

Este modelo se puede extender a fijar una cota superior a la cantidad de vehículos a utilizar en la solución, para ello debemos agregar la siguiente restricción al modelo.

$$\sum_{r \in R} \sum_{j=0}^{n+1} x_{0jr} \leq |V| \quad \forall r \in R \tag{16}$$

3.2.4. Vehicle Routing Problem with Pickup and Delivery and Time Windows

En el problema de recolección y entrega con ventanas de tiempo[28] (VRPPDTW, por sus siglas en inglés), se deben atender solicitudes de transporte que involucran puntos de recolección y entrega. Cada ubicación de recolección debe ser visitada antes que la correspondiente ubicación de entrega (restricción de precedencia). Además, la capacidad de los vehículos no debe ser excedida en ningún momento (restricción de capacidad), y el servicio a cada cliente debe comenzar dentro del intervalo de tiempo asignado a dicha solicitud (restricción de ventana de tiempo).

Este modelo es una extensión del VRPPD (Vehicle Routing Problem with Pickup and Delivery) que incluye restricciones adicionales en forma de ventanas de tiempo. El objetivo[34] es encontrar las rutas óptimas para los vehículos que no solo minimicen el costo total (como en el VRPPD) sino que también respeten las ventanas de tiempo para cada punto de recogida y entrega.

Modelo Propuesto

El modelo matemático del VRPPDTW con múltiples eventos dinámicos en un entorno real se presenta a continuación[35]:

Parámetros

- K: Número total de vehículos.
- C: Máxima duración del horizonte de planificación.

- Q_k: Capacidad máxima de cada vehículo.
- D: Distancia máxima de conducción de cada vehículo.
- t: Punto de tiempo en el cual se optimiza la planificación de la ruta.
- $N(t)$: Conjunto de ubicaciones junto con solicitudes de clientes no aceptadas, nuevas solicitudes dinámicas y modificaciones de solicitudes.
- N^+: Conjunto de nodos de recogida.
- N^-: Conjunto de nodos de entrega.
- d_{ij}: Distancia entre cada par de nodos (v_i, v_j).
- $s_k(t)$: Velocidad del vehículo k en el tiempo t.
- $t_{ij}^k(t)$: Tiempo de viaje del vehículo k desde el nodo v_i al nodo v_j en el tiempo t.
- $D_k(t)$: Distancia acumulada recorrida por el vehículo k después de salir del depósito en el tiempo t.
- $Q_k(t)$: Capacidad restante del vehículo k en el tiempo t.
- $W_p(t)$: Conjunto de ubicaciones de todos los vehículos en la red en el tiempo t.
- β: Costo fijo por vehículo, que incluye costos de depreciación, mantenimiento, etc.
- c: Costo unitario de viaje del vehículo.
- α: Grado de urgencia.
- ssi: Tiempo de operación en el nodo v_i.
- q_i: Cantidad de demanda del nodo v_i.
- $[e_i, l_i]$: Ventana de tiempo de servicio en el nodo v_i.

Variables de Decisión

- x_{ij}^k: Variable binaria que indica si el vehículo k es enviado desde v_i a v_j.
- y_j^k: Carga del vehículo k después de servir el nodo v_j.

- a_j^k: Tiempo necesario para que el vehículo k llegue al nodo v_j.
- w_j^k: Tiempo de espera del vehículo k en el nodo v_j.
- de_i^k: Tiempo total que toma hasta que el vehículo k deja el nodo v_i.
- tl_i^k: Retraso con el que el vehículo k llega al nodo v_i.
- d_i^k: Distancia de viaje después de que el vehículo k ha servido el nodo v_i.
- u_k: Variable binaria que indica si el vehículo k es utilizado.
- n_t: Número de vehículos planificados para uso en el tiempo t.

Función Objetivo

$$\text{mín } Z = \sum_{i \in W_{up0}(t)} \sum_{j \in W_{u0}(t)} \sum_{k=1}^{K} cd_{ij} x_{ij}^k + \sum_{i \in W_{up0}(t)} \sum_{k=1}^{K} \alpha tl_i^k + \beta n_t \tag{17}$$

sujeto a:

$$\sum_{k=1}^{K} \sum_{j \in N(t)} x_{ij}^k = 1, \quad \forall i \in N(t) \tag{18}$$

$$\sum_{k=1}^{K} \sum_{j \in N(t)} x_{ij}^k = 1, \quad \forall i \in W_{p0}(t) \tag{19}$$

$$\sum_{i \in N(t) \cup W_p(t)} x_{i0}^k = 1, \quad \forall k = 1, \ldots, K \tag{20}$$

$$\sum_{i \in N(t)} x_{ih}^k - \sum_{j \in N(t)} x_{hj}^k = 0, \quad \forall h \in N(t), \forall k = 1, \ldots, K \tag{21}$$

$$\sum_{l \in N(t)} x_{li}^k z_{ij} - \sum_{p \in N(t)} x_{pj}^k z_{ij} = 0, \quad \forall i, j \in N(t), \forall k = 1, \ldots, K \tag{22}$$

$$y_j^k \leq Q, \quad \forall j \in N(t), \forall k = 1, \ldots, K \tag{23}$$

$$y_j^k = Q_k(t), \quad \forall j \in W_p(t), \forall k = 1, \ldots, K \tag{24}$$

$$x_{ij}^k (y_j^k - y_i^k - q_j) = 0, \quad \forall i, j \in N(t), \forall k = 1, \ldots, K \tag{25}$$

$$x_{ij}^k (de_i^k + t_{ij}^k) \leq a_j^k, \quad \forall i, j \in N(t), \forall k = 1, \ldots, K \tag{26}$$

$$de_i^k = \text{máx}\{a_j^k, e_i\} + ssi, \quad \forall i \in N(t) \tag{27}$$

$$z_{ij} a_i^k \leq a_j^k, \quad \forall i, j \in N(t), \forall k = 1, \ldots, K \tag{28}$$

$$x_{i0}^k (de_i^k + t_{i0}^k) \leq C, \quad \forall i \in N(t), \forall k = 1, \ldots, K \tag{29}$$

La ecuación (18) asegura que cada solicitud i en el conjunto $N(t)$ sea atendida exactamente una vez por alguno de los vehículos disponibles. Esto garantiza que todas las demandas de los clientes sean cumplidas sin duplicación de servicios.

La ecuación (19) es similar a la anterior, pero se enfoca en las ubicaciones de recogida i en $W_{p0}(t)$, asegurando que cada recogida se realice exactamente una vez. Esta restricción es crucial para manejar correctamente las solicitudes de recogida en el sistema.

La ecuación (20) asegura que cada vehículo salga del depósito al inicio del período de despacho. Esto establece un punto de partida claro para cada vehículo y facilita la planificación de las rutas.

La ecuación (21) garantiza que si un vehículo llega a un nodo h, también debe salir de ese

nodo, manteniendo la continuidad de la ruta. Esta restricción es esencial para asegurar que los vehículos no queden atrapados en ningún nodo y que las rutas sean continuas y lógicas.

La ecuación (22) asegura que si un vehículo recoge en un nodo i, también debe entregar en el nodo correspondiente j. Esto garantiza que las operaciones de recogida y entrega estén emparejadas correctamente, lo cual es fundamental para la consistencia del servicio.

La ecuación (23) limita la carga del vehículo k después de servir el nodo j a la capacidad máxima Q_k. Esto asegura que los vehículos no excedan su capacidad de carga, manteniendo así la viabilidad operativa y la seguridad.

La ecuación (24) establece la carga inicial del vehículo k en los nodos de recogida j como $Q_k(t)$. Esta restricción es importante para reflejar la cantidad de carga que los vehículos llevan al comienzo del proceso de servicio.

La ecuación (25) asegura que el flujo de carga del vehículo k sea consistente con la demanda, ajustando la carga del vehículo después de servir el nodo j en función de la carga antes de servir el nodo i y la demanda q_j. Esto es crucial para mantener el balance de carga a lo largo de la ruta.

La ecuación (26) garantiza que los tiempos de llegada a_j^k sean consistentes con los tiempos de servicio y los tiempos de viaje entre nodos, asegurando que los vehículos lleguen a tiempo a cada nodo de servicio.

La ecuación (27) establece el tiempo de salida del nodo i como el máximo entre el tiempo de llegada a_j^k y el tiempo de inicio de servicio e_i, más el tiempo de operación ssi. Esto garantiza que los vehículos no salgan antes de completar el servicio requerido.

La ecuación (28) asegura que los vehículos lleguen dentro de la ventana de tiempo permitida, restringiendo los tiempos de llegada a_i^k y a_j^k para cumplir con las ventanas de tiempo de los nodos i y j.

Finalmente, la ecuación (29) restringe el tiempo de regreso al depósito para que sea dentro del horizonte de planificación C. Esto asegura que todas las rutas se completen dentro del tiempo disponible, evitando operaciones fuera del horario planificado.

3.3. Soluciones Exactas VRP

El Problema de Ruteo de Vehículos es una generalización del Problema del Viajante (TSP) y es significativamente más desafiante de resolver exactamente. A lo largo de los años, se han desarrollado varios algoritmos exactos para abordar el VRP, entre los cuales se destacan Branch and Bound y Dynamic Programming. En esta sección, se explican en detalle distintos métodos y se presentan los modelos matemáticos subyacentes junto a un pseudocódigo de cada algoritmo para su implementación.

El Método Simplex, desarrollado por Dantzig en 1947[36], dio origen a técnicas como Branch and Bound, Column Generation y Branch and Cut, que facilitan la búsqueda de soluciones en algoritmos de optimización.

En 1987, Laporte y Norbert clasificaron los métodos exactos en tres grupos: *Direct Tree Search Methods*, *Dynamic Programming* e *Integer Linear Programming*[37].

- *Direct Tree Search Methods* utilizan un árbol Branch and Bound para construir rutas secuenciales. Ejemplos incluyen el algoritmo de Baker [38] y el trabajo de Christofides y Eilon [39].
- *Dynamic Programming* se enfoca en relajaciones para problemas como el TSP con ventanas de tiempo, con aportes de Christofides et al. [39] y Eilon et al. [40].
- *Integer Linear Programming* usa métodos como Branch and Bound para resolver problemas de programación lineal. Ejemplos incluyen el algoritmo de Dastghaibifard[41] para problemas de ruteo de vehículos (2003) y el enfoque Branch and Cut de Ralphs[42].

3.3.1. Branch and Bound

Descripción del Algoritmo

En el artículo presentado por Little, Murty, Sweeney y Karel[43] se plantea un algoritmo desarrollado específicamente para resolver el TSP basado en la técnica de Branch and Bound. Este método es conocido por su eficiencia en la reducción del espacio de búsqueda mediante el uso de límites inferiores para eliminar subproblemas que no pueden contener la solución óptima.

El algoritmo de Little et al. sigue estos pasos[44]:

- **Inicialización:** Comienza con una matriz de costos que representa el grafo de conexiones entre los diferentes nodos del sistema.
- **Reducción de Matriz:** Reduce la matriz de costos para obtener una cota inferior inicial. Esto se logra restando el valor mínimo de cada fila y columna. Definimos este valor como $\overline{\gamma}$.
- **Ramificación:** Se generan todos los hijos del nodo en expansión antes de que cualquier otro nodo vivo pase a ser el nuevo nodo en expansión. Al iniciar el árbol por primera vez el nodo raíz representa el lugar de salida.
- **Cálculo de Límite Inferior:** Se elige un nodo a expandir (nodo expansión) entre los nodos vivos. Vuelve a ajustarse la matriz y se repite el proceso con el siguiente nivel de nodos vivos.
- **Podas y Selección:** Descarta los nodos vivos con cotas inferiores mayores que la mejor solución conocida. Estos nodos se etiquetan como nodos muertos y si existe un nodo vivo con un Lower Bound menor a la mejor solución conocida, se convierte al nodo vivo en uno de expansión y se recalcula la solución.

Algorithm 1 Algoritmo de Little et al. para TSP

```
1: procedure LITTLE(Matriz C)
2:     Reducir la matriz C para obtener la cota inferior inicial LB
3:     Inicializar el árbol de búsqueda con el nodo raíz
4:     while nodos vivos en el árbol de búsqueda  do
5:         Seleccionar el nodo con el menor costo de traslado (nodo expansión)
6:         Reducir la matriz para el nodo en expansión y calcular el LB parcial
7:         if el nodo es una solución completa then
8:             Actualizar la mejor solución con LB
9:             Podar los nodos vivos con LB mayor que la mejor solución actual
10:        else
11:            Ramificar el siguiente nivel con los nodos vivos aún no recorridos y actualizar
    nodo expansión
12:        end if
13:    end while
14:    return la mejor solución encontrada
15: end procedure
```

Otro algoritmo basado en Branch and Bound para VRP fue propuesto por Christofides y Eilon[39]. Este método se basa en el algoritmo de ramificación y poda de Little et al., donde el recorrido final pasa por cada cliente y por el depósito solo una vez. El problema puede formularse como un $N-TSP$ eliminando el depósito real y reemplazándolo por N depósitos artificiales, todos ubicados en la misma posición. Viajar de un depósito artificial a otro está prohibido estableciendo la distancia entre cualquier par de depósitos igual a infinito, como se muestra en la matriz de costos en la Figura 3.2, y ahora se busca una solución a esta matriz de costos, sujeta a las restricciones de capacidad de los vehículos.

El número de depósitos artificiales N es igual al número de vehículos empleados en la solución final, y este número tiene un límite inferior determinado por la capacidad del vehículo, a saber:

$$N \geq \frac{\sum_{i=1}^{n} q_i}{C}$$

donde q_i es la demanda para el cliente i $(i = 1, 2, .., n)$ y C es la capacidad del vehículo. Los problemas pueden ser resueltos para varios valores de N y se elige la mejor solución.

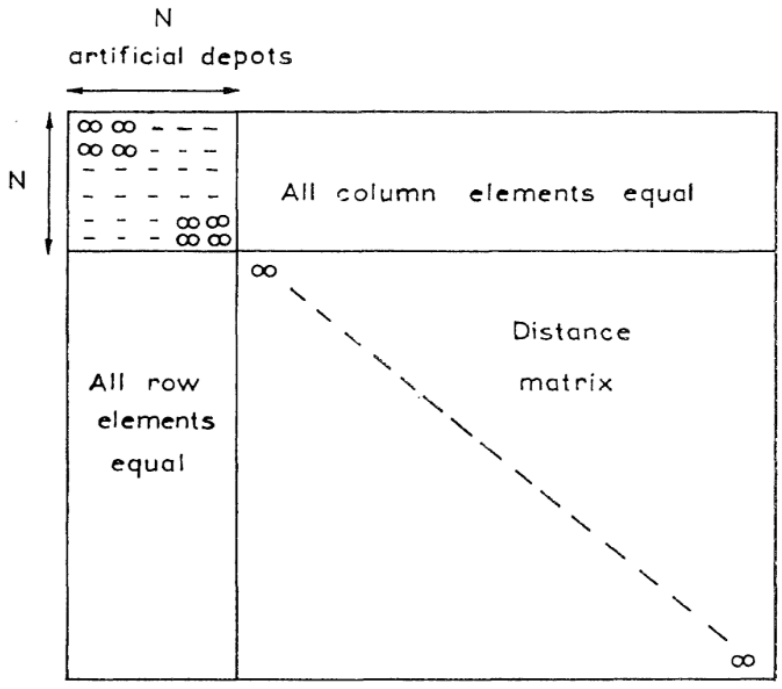

Figura 3.2: Matriz para el problema de ruteo de vehículos[39].

El algoritmo de Christofides implica las siguientes etapas:

- **Extensión del Grafo**: Se añaden depósitos artificiales al grafo para transformar el VRP en un problema de múltiple TSP.
- **Resolución del** $N - TSP$: Se resuelve un problema de múltiple TSP en el grafo extendido (Little et al).
- **Inicialización del Árbol de Búsqueda**: Se comienza con un nodo raíz que representa el problema completo.
- **Selección y Ramificación**: Se selecciona un nodo para ramificar, dividiendo el problema en subproblemas más pequeños.
- **Cálculo de Límites**: Se calculan límites inferiores para los subproblemas y se descartan aquellos que no pueden mejorar la mejor solución encontrada hasta el momento.
- **Actualización de la Mejor Solución**: Si se encuentra una solución factible mejor que la actual, se actualiza la mejor solución.

Algorithm 2 Branch and Bound para VRP

```
1: procedure BranchAndBound(Grafo G, Depósitos m)
2:     Extender G añadiendo m − 1 depósitos artificiales
3:     Establecer costos de arco inter-depósito a infinito
4:     Resolver un m-TSP en el grafo extendido
5:     Inicializar el árbol de búsqueda con el nodo raíz
6:     while hayan nodos en el árbol de búsqueda do
7:         Seleccionar un nodo para ramificar
8:         if el nodo representa una solución factible then
9:             Actualizar la mejor solución encontrada
10:        else
11:            Calcular un límite inferior para el nodo
12:            if el límite es menor que la mejor solución actual then
13:                Ramificar en el nodo añadiendo nodos hijos al árbol de búsqueda
14:            end if
15:        end if
16:    end while
17:    return mejor solución encontrada
18: end procedure
```

3.3.2. Programación Dinámica

La Programación Dinámica (DP) es otro enfoque exacto para resolver VRP. Involucra descomponer el problema en subproblemas más simples y resolverlos recursivamente. Eilon, Watson-Gandy y Christofides [40] formularon el VRP como un problema dinámico.

El algoritmo DP puede ser formulado matemáticamente como sigue:

$$f_k(U) = \begin{cases} c(U) & \text{si } k = 1 \\ \text{mín}_{U^* \subset U}\{f_{k-1}(U \setminus U^*) + c(U^*)\} & \text{si } k > 1 \end{cases} \tag{30}$$

(31)

Donde $f_k(U)$ es el costo mínimo usando k vehículos para entregar al subconjunto U de vértices. El costo de la solución es $f_m(V \setminus \{0\})$, y la partición óptima corresponde a los subconjuntos que optimizan el valor.

Descripción del Algoritmo

En la formulación de DP, el VRP se resuelve encontrando el costo óptimo de una sola ruta a través de subconjuntos de vértices y luego particionando los vértices entre múltiples vehículos.

El proceso de DP implica las siguientes etapas:

- Inicialización de la Tabla de Costos: Se inicializa una tabla de costos $c(S)$ para todos los subconjuntos S de vértices.
- Inicialización de la Tabla de Estados: Se inicializa una tabla de estados $f_k(U)$ para todos los subconjuntos U de vértices y k vehículos.
- Cálculo de Costos de Subconjuntos: Para cada subconjunto U de vértices, se calcula $c(U)$.
- Iteración sobre Vehículos: Para cada número de vehículos k, se actualiza la tabla de estados considerando la partición óptima de los vértices.

Pseudocódigo

Algorithm 3 Programación Dinámica para VRP

procedure DynamicProgramming(Grafo G, Vehículos m)
    Inicializar tabla de costos $c(S)$ para todos los subconjuntos S de vértices
    Inicializar tabla de estados $f_k(U)$ para todos los subconjuntos U de vértices y k vehículos
    for cada subconjunto U de vértices **do**
        Calcular $c(U)$
    end for
    for $k = 2$ a m **do**
        for cada subconjunto U de vértices **do**
            $f_k(U) = \min_{U^* \subset U}\{f_{k-1}(U \setminus U^*) + c(U^*)\}$
        end for
    end for
    return $f_m(V \setminus \{0\})$
end procedure

3.4. Heurísticas

En el ámbito de la Investigación Operativa, una heurística se define como un procedimiento diseñado para encontrar soluciones de alta calidad en un tiempo computacional razonable, especialmente útil cuando los métodos exactos resultan inviables debido a la complejidad o escala del problema[45].

Las heurísticas son técnicas de búsqueda que proporcionan soluciones aproximadas sin garantizar la óptima global. En VRP, estas estrategias pueden dividirse en métodos constructivos y métodos de mejora, permitiendo generar soluciones iniciales y refinarlas progresivamente.

A su vez, las heurísticas constructivas pueden dividirse en métodos secuenciales y paralelos. Los procedimientos secuenciales construyen una ruta a la vez hasta que todos los clientes han sido asignados a una ruta. Los procedimientos paralelos se caracterizan por la construcción simultánea de varias rutas. Estas rutas pueden formarse libremente o su número puede estar previamente fijado.

En este capítulo se describen técnicas, métodos, algoritmos y estrategias que permiten ser aplicados para la búsqueda de soluciones al problema VRP.

3.4.1. Savings Heuristic

Introducido por Clarke and Wright[46] en 1964, el algoritmo de ahorros es una Heurística constructiva que tiene como objetivo reducir la distancia total recorrida combinando rutas individuales en una sola ruta más eficiente, generando un ahorro. Se comienza con una solución en la que cada cliente tiene su propio recorrido individual desde y hacia el depósito (una ruta por cliente) y se calcula cuánto se ahorraría en distancia si dos clientes fueran atendidos en una misma ruta en lugar de rutas separadas.

Si en una solución existen dos rutas separadas, una de la forma $(0, \ldots, i, 0)$ y otra como $(0, j, \ldots, 0)$, es posible combinarlas en una única ruta $(0, \ldots, i, j, \ldots, 0)$, como se muestra en la Figura 3.3.

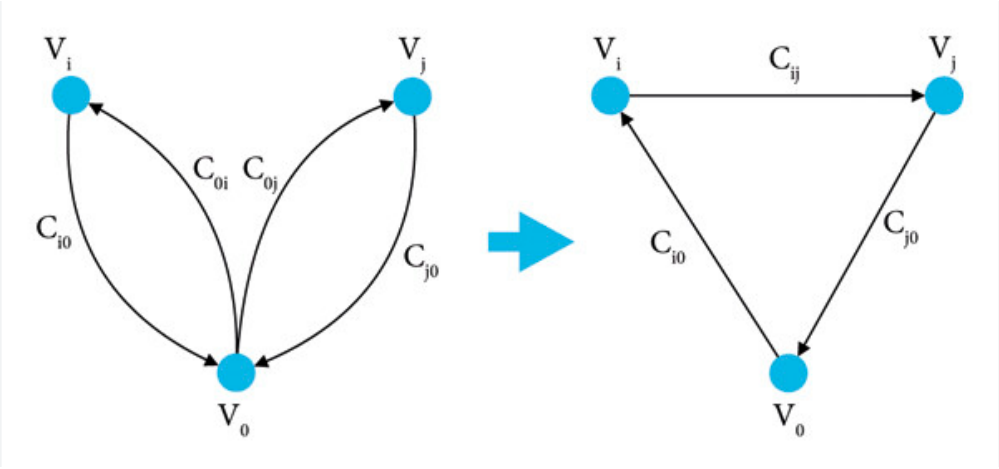

Figura 3.3: Ejemplo de heurística de ahorro.

El ahorro en distancia generado por esta unión se calcula mediante la siguiente fórmula:

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \tag{32}$$

Esto se debe a que, en la nueva solución, los arcos $(i, 0)$ y $(0, j)$ ya no se utilizan, mientras que se introduce el arco directo (i, j), optimizando así el recorrido total.

El algoritmo de Clarke and Wright tiene dos variantes: una paralela, que procesa todas las rutas simultáneamente optimizando globalmente, y una secuencial, que las construye una por una de manera progresiva. La elección entre ambas depende del equilibrio entre eficiencia y simplicidad en la implementación[47].

Algoritmo de Ahorros (Versión paralela)

- **Paso 1 (inicialización).** Para cada cliente i construir la ruta $(0, i, 0)$.
- **Paso 2 (cálculo de ahorros).** Calcular s_{ij} para cada par de clientes i y j.
- **Paso 3 (mejor unión)**. Sea $s_{i^*j^*} = \max s_{ij}$, donde el máximo se toma entre los ahorros que aún no han sido considerados. Sean r_{i^*} y r_{j^*} las rutas que contienen a los clientes i^* y j^*, respectivamente. Si i^* es el último cliente de r_{i^*} y j^* es el primer cliente de r_{j^*}, y la combinación de r_{i^*} y r_{j^*} es factible, se deben fusionar ambas rutas. Luego, eliminar $s_{i^*j^*}$ de futuras consideraciones. Si aún quedan ahorros por examinar, volver al paso 3; de lo contrario, finalizar el procedimiento.

Algoritmo de Ahorros (Versión secuencial)

- **Paso 1 (Inicialización).** Para cada cliente i, construir la ruta $(0, i, 0)$.
- **Paso 2 (Cálculo de ahorros).** Calcular s_{ij} para cada par de clientes i y j.
- **Paso 3 (Selección).** Si todas las rutas fueron consideradas, finalizar el procedimiento. En caso contrario, seleccionar una ruta que aún no haya sido procesada.
- **Paso 4 (Extensión).** Sea $(0, i, \ldots, j, 0)$ la ruta actual. Si no existe ningún ahorro que involucre a i o j, regresar al paso 3. Sea s_{k^*i} (o s_{jl^*}) el ahorro máximo que contenga a i (o j). Si k^* (o l^*) es el último (o primer) cliente de su ruta y la combinación de esa ruta con la actual es factible, se debe realizar la fusión. Luego, eliminar s_{k^*i} (o s_{jl^*}) de futuras consideraciones y regresar al paso 4.

Las soluciones obtenidas con el Algoritmo de Ahorros pueden, en general, ser mejoradas mediante operadores de búsqueda local como el algoritmo 3-opt [48].

3.4.2. Insertion Heuristic

Las heurísticas de inserción son métodos constructivos que generan una solución agregando clientes a las rutas de manera progresiva. En cada iteración, se parte de una solución parcial en la que solo un subconjunto de clientes ha sido asignado a rutas, y se selecciona un cliente no visitado para insertarlo en la mejor posición posible dentro de la solución existente. Este proceso se repite iterativamente hasta que todos los clientes hayan sido asignados a una ruta, asegurando así una construcción eficiente y adaptativa de la solución[47].

Solomon

Solomon[1] extiende la heurística de inserción para contemplar algunas restricciones adicionales como ventanas de tiempo, capacidad y flota heterogénea. Todas se basan en extensiones de las funciones de costo para incorporar tanto las distancias como el tiempo.

Los métodos de inserción son una estrategia efectiva en problemas con restricciones de ventanas de tiempo. Estas heurísticas construyen rutas de manera incremental, insertando

clientes en posiciones que minimicen el incremento en el costo total y aseguren la factibilidad temporal. Solomon[1] destaca que estos métodos son especialmente útiles en escenarios donde las restricciones temporales limitan las opciones de ruteo. Existen diferentes variantes:

- **Inserción simple**: Se evalúan todas las posibles posiciones en la ruta y se elige la que minimiza el incremento en la distancia total.
- **Inserción por urgencia**: Se priorizan los clientes con ventanas de tiempo más ajustadas, asegurando que sean atendidos sin violar restricciones temporales.
- **Inserción ponderada**: Combina múltiples factores, como la distancia, el tiempo de espera y las restricciones de ventana de tiempo, utilizando una función de costo.

Solomon realiza el siguiente planteo matemático para la heurística de inserción: Sea i un cliente perteneciente al conjunto $\{1, ..., n\}$. A cada cliente se le asigna un tiempo de servicio s_i, el cual comienza en el instante b_i, siempre que este se encuentre dentro de una ventana de tiempo $[e_i, l_i]$. Esta ventana está definida por el instante más temprano e_i y el más tardío l_i, que representa el límite impuesto por el cliente para iniciar el servicio.

Si un vehículo se traslada directamente del cliente i al cliente j y llega antes del tiempo permitido en j, deberá esperar. En este caso, el tiempo de inicio del servicio en j viene dado por:

$$b_j = \text{máx}(e_j, b_i + s_i + t_{ij}) \tag{33}$$

donde t_{ij} representa el tiempo de viaje directo entre i y j. Es importante destacar que los valores de b_i, para $i = 1, ..., n$, son variables de decisión [1].

El costo asociado a un viaje directo entre los clientes i y j se define como:

$$c_{ij} = \rho_1 d_{ij} + \rho_2 (b_j - b_i) \tag{34}$$

donde $\rho_1 \geq 0$, $\rho_2 > 0$, y d_{ij} es la distancia entre ambos clientes. En particular, si $\rho_1 = 0$ y $\rho_2 = 1$, el objetivo se reduce a minimizar el tiempo total de la ruta.

Además, suponemos que el número de vehículos disponibles no está restringido, por lo que la flota se determina simultáneamente con la selección óptima de rutas y horarios. Los

vehículos parten desde un depósito, representado por el nodo $i = 0$, en un intervalo de tiempo $[e_0, l_0]$. Los tiempos de salida desde el depósito también se consideran variables de decisión.

La efectividad de los métodos heurísticos para este problema depende en gran medida de la manera en que se manejan las restricciones de ventanas de tiempo en la construcción de soluciones[49]. Como nos enfocaremos en procedimientos de construcción de rutas, analizaremos primero las condiciones necesarias y suficientes para garantizar la viabilidad temporal al insertar un cliente u entre los clientes i_{p-1} e i_p, con $1 < p < m$, dentro de una ruta factible parcialmente construida:

$$(i_0, i_1, i_2, \ldots, i_m), \quad i_0 = i_m = 0, \tag{35}$$

donde los tiempos de inicio del servicio, b_i, para $0 \leq i \leq m$, son conocidos. Inicialmente, asumimos que cada vehículo parte del depósito en el instante más temprano permitido, e_0. Sin embargo, una vez establecidas las rutas completas, los tiempos de salida pueden ajustarse individualmente para cada vehículo con el fin de minimizar tiempos de espera innecesarios.

Denotemos como b_{new} el nuevo tiempo de inicio del servicio en i_p tras insertar el cliente u. Asimismo, definimos w_{ir} como el tiempo de espera en el cliente i_r para $p < r < m$.

El desplazamiento hacia adelante (Push Forward, PF) en i_p se expresa como:

$$PF_{i_p} = b_{i_p}^{new} - b_{i_p} \geq 0 \tag{36}$$

Además, para los clientes posteriores:

$$PF_{i_{r+1}} = \text{máx}(0, PF_{i_r} - w_{i_{r+1}}), \quad p \leq r < m \tag{37}$$

Si se tiene $PF_{i_p} > 0$, algunos clientes i_r, con $p < r < m$, podrían volverse inviables. Para verificar su viabilidad temporal, evaluamos secuencialmente cada cliente hasta encontrar uno que cumpla alguna de las siguientes condiciones:

(1) $PF_{i_r} = 0$,

(2) el cliente resulta inviable en términos de tiempo

(3) se han examinado todos los clientes en el rango $p \leq r \leq m$.

De este modo, las condiciones necesarias y suficientes para garantizar la viabilidad temporal al insertar un cliente u entre i_{p-1} e i_p, con $1 \leq p \leq m$, dentro de una ruta factible parcialmente construida $(i_0, i_1, i_2, \ldots, i_m)$, donde $i_0 = i_m = 0$, son:

$$b_u \leq l_u, \quad \text{y} \quad b_{i_r} + PF_{i_r} \leq l_{i_r}, \quad p \leq r \leq m. \tag{38}$$

Además, dado que $i_m = 0$, cualquier cliente que no permita que el vehículo regrese al depósito dentro del tiempo programado no sea agregado a la ruta parcial.

Para evaluar la mejor posición de inserción de un cliente u entre dos clientes i y j en una ruta parcial (i, j), se utilizan las siguientes funciones de costo:

$c_1(i, u, j)$: Evalúa el incremento en la distancia y el tiempo al insertar el cliente u entre los clientes i y j:

$$c_1(i, u, j) = \alpha_1 c_{11}(i, u, j) + \alpha_2 c_{12}(i, u, j) + \alpha_3 c_{13}(i, u, j) \tag{39}$$

donde $\alpha_1, \alpha_2, \alpha_3$ son parámetros de ponderación que cumplen $\alpha_1 + \alpha_2 + \alpha_3 = 1$ y $\alpha_1 \geq 0$, $\alpha_2 \geq 0$, $\alpha_3 \geq 0$.

$c_{11}(i, u, j)$: Evalúa el incremento en la distancia de la ruta debido a la inserción de u:

$$c_{11}(i, u, j) = d_{iu} + d_{uj} - \lambda d_{ij} \tag{40}$$

donde d_{iu} y d_{uj} representan las distancias entre los clientes respectivos y λ ajusta la penalización por desviación de la ruta original.

$c_{12}(i, u, j)$: Considera el impacto temporal de la inserción en la programación del servicio:

$$c_{12}(i, u, j) = b'_j - b_j \tag{41}$$

donde b'_j es el nuevo tiempo de inicio de servicio en el cliente j después de insertar a u.

$c_{13}(i, u, j)$: Evalúa la urgencia del servicio para el cliente u, favoreciendo clientes con menor tiempo disponible:

$$c_{13}(i, u, j) = l_u - b_u \tag{42}$$

donde l_u es el tiempo límite para iniciar el servicio en u y b_u el tiempo en que efectivamente se programaría el servicio.

Por otro lado, para seleccionar el cliente óptimo a insertar en la ruta se utiliza la siguiente función, la cual pondera tanto la distancia como el impacto en el tiempo de ruta total:

$$c_2(i,u,j) = \beta_1 R_d(u) + \beta_2 R_t(u) \tag{43}$$

donde $\beta_1 + \beta_2 = 1$, $\beta_1 \geq 0$ $\beta_2 \geq 0$ son parámetros de ponderación y $R_d(u)$ y $R_t(u)$ representan el impacto en la distancia y el tiempo total de la ruta si u es insertado.

Potvin y Rousseau (VRPTW)

La heurística propuesta por Potvin y Rousseau [50] para el *VRPTW* es una versión paralela de la heurística de inserción secuencial desarrollada por Solomon. En este enfoque, la cantidad de rutas en la solución se determina al inicio y permanece fija durante toda la ejecución del algoritmo. Para establecer el número de rutas, primero se aplica una de las heurísticas de inserción secuencial de Solomon, y a partir del resultado obtenido se genera una nueva solución con la misma cantidad de rutas.

Además, para inicializar las rutas en la nueva solución, se selecciona de cada una de las rutas generadas por el algoritmo de Solomon el cliente más alejado del depósito, asignándolo como el primer cliente de una nueva ruta. De esta manera, el algoritmo de Solomon no solo define la cantidad de rutas a crear, sino que también se emplea para determinar la estructura inicial de dichas rutas.

Durante cada iteración, se debe seleccionar qué cliente incorporar a la solución y en qué posición dentro de la ruta ubicarlo. Esto implica decidir tanto la ruta específica como la ubicación exacta del cliente dentro de ella. Al igual que en el algoritmo de Solomon, se emplean dos métricas para determinar la mejor inserción. En particular, se introduce la medida $c_{1r}(v_i, w)$, que es una variante de la métrica c_1 utilizada en los algoritmos de Solomon.

Las siguientes ecuaciones definen la medida $c_{1r}(v_i, w)$:

$$c_{1r}(v_i,w) = \alpha_1 c_{11r}(v_i,w) + \alpha_2 c_{12r}(v_i,w) \tag{44}$$

$$c_{11r}(v_i,w) = d_{ik} + d_{kj} - \mu d_{ij} \tag{45}$$

$$c_{12r}(v_i, w) = b_{jk} - b_j \tag{46}$$

Es decir, se trata de la suma ponderada del aumento en la distancia y el retraso causados por la inserción del cliente w entre los clientes consecutivos v_i y v_{i+1} en la ruta r. Cabe destacar que existe un valor de esta medida para cada ruta. A continuación, para cada cliente w, se determina la mejor opción:

$$c^*_{1r^*}(v^*_i, w) = \min c_{1r}(v_i, w) \tag{47}$$

donde el mínimo se toma de todas las posibles rutas r y todos los pares de nodos consecutivos en dicha ruta.

$$c_2(w) = \sum_{r \neq r^*} (c_{1r}(v_i, w) - c^*_{1r^*}(v^*_i, w)) \tag{48}$$

Esta medida cuantifica la urgencia de insertar al cliente w en la solución. Para ello, compara el costo de la mejor inserción posible con el de las demás alternativas: cuanto mayor sea la diferencia, más urgente será considerar al cliente. Es decir, valores altos de $c_2(w)$ indican que existe una opción claramente superior a las demás, por lo que el cliente debería priorizarse. A partir de esta evaluación, se selecciona el cliente w^* como:

$$w^* = \max_w c_2(w) \tag{49}$$

y se inserta en la ruta r^* en la posición dada por c_{1r^*}.

Si la inserción no es factible, c_1 toma un valor grande L. Si el cliente w puede ser insertado en N_w rutas, el valor de $c_2(w)$ será cercano a $(N - N_w)L$. Así, el cliente con menos alternativas tiene mayor prioridad. En caso de que dos clientes tengan la misma cantidad de opciones, se discrimina por el costo de inserción.

El algoritmo utiliza los parámetros α_1 y α_2 en c_{1r}. Se ejecuta varias veces ajustando estos parámetros y disminuyendo N hasta encontrar soluciones factibles.

Mole & Jameson

La heurística propuesta por Mole y Jameson [51] es un enfoque secuencial para la construcción de rutas que extiende el criterio de ahorro clásico de Clarke y Wright, introducien-

do una versión generalizada del mismo y permitiendo mayor flexibilidad en la inserción de clientes. Esta heuristica considera dos medidas para determinar cuál es el próximo cliente a insertar en la ruta de la solución parcial.

Una primera medida consiste en calcular, para cada cliente no visitado, la mejor posición para insertarlo en la ruta actual, siendo esta ruta:

$$(v_0, v_1, \ldots, v_t, v_{t+1}) \quad \text{donde } v_0 = v_{t+1} \tag{50}$$

Si w es el cliente considerado para ser insertado en la ruta, su *costo de inserción* se define como:

$$c_1(v_i, w) = \begin{cases} c(v_i, w) + c(w, v_{i+1}) - \lambda c(v_i, v_{i+1}), & \text{si la inserción} \\ (v_0, v_1, \ldots, v_i, w, v_{i+1}, \ldots, v_t, v_{t+1}) \text{ es factible} & \\ \infty, & \text{si la inserción no es factible} \end{cases} \tag{51}$$

La mejor posición para insertar al cliente w en la ruta actual está dada por:

$$i(w) = \arg \min_{i=0,\ldots,t} c_1(i, w) \tag{52}$$

Si se utilizara solamente la medida c_1 para decidir el próximo cliente a insertar, es probable que los clientes lejanos al depósito no sean considerados sino hasta las últimas iteraciones del algoritmo, es decir, cuando sean las únicas alternativas factibles.

Por lo tanto, es necesario utilizar un incentivo adicional para la inserción de clientes lejanos al depósito. Esta medida se define como:

$$c_2(v_i, w) = \mu c_{0w} - c_1(v_i, w) \tag{53}$$

donde c_{0w} es la distancia entre el depósito y el cliente w, y μ es un parámetro de ponderación.

En cada iteración, se busca el cliente que maximiza la medida c_2 (llamada *medida de urgencia*), y se lo inserta en la posición dada por el mínimo valor de c_1.

Además de las medidas anteriores, debe considerarse la factibilidad de las inserciones.

Cuando ninguna inserción es factible y aún quedan clientes sin visitar, se selecciona un cliente para comenzar una nueva ruta.

El algoritmo es el siguiente.

- **Paso 1: Inicio de ruta.** Si aún quedan clientes sin asignar, se selecciona uno (por ejemplo, el más lejano al depósito) y se inicia una nueva ruta $r = (0, w, 0)$.
- **Paso 2: Inserción.** Para cada cliente no visitado, calcular c_1 y c_2. Insertar el cliente w^* que maximiza c_2 en la mejor posición $i(w^*)$.
- **Paso 3: Optimización local.** Aplicar un procedimiento 3-opt sobre la ruta actual para mejorar su secuencia interna.
- **Paso 4.** Repetir desde el paso 2 hasta que no haya más inserciones factibles. Luego, volver al paso 1.

Christofides, Mingozzi y Toth

El algoritmo propuesto por Christofides, Mingozzi y Toth [52] es una heurística de inserción en paralelo diseñada para resolver problemas de ruteo de vehículos con restricciones de capacidad y ventanas de tiempo. Esta heurística se estructura en dos fases: la primera se enfoca en la determinación de la cantidad de rutas y sus clientes iniciales; la segunda se encarga de construir y completar las rutas a partir de dichas asignaciones.

Fase 1: Determinación de rutas iniciales En la primera fase, se genera una solución parcial aplicando una heurística de inserción secuencial con el objetivo de obtener rutas compactas. Sin embargo, no se conserva el orden de los clientes en las rutas; solo se retienen los clientes iniciales y la cantidad total de rutas resultantes.

Para inicializar una ruta r_k, se selecciona un cliente v_k entre los no visitados. Luego, se define el costo de insertar un cliente w en la ruta que contiene a v_k como:

$$\delta_{w,v_k} = \begin{cases} c_{0w} + \lambda c_{w,v_k} & \text{si la inserción es factible} \\ \infty & \text{en caso contrario} \end{cases} \tag{54}$$

donde c_{0w} es la distancia desde el depósito hasta el cliente w, y λ es un parámetro que pondera el costo de conectar w con el cliente inicial de la ruta. Se insertan en la ruta los clientes no visitados con menor valor de δ, hasta que no sea posible realizar más inserciones factibles. Luego, se crea una nueva ruta o se finaliza la fase.

Fase 1 - Algoritmo de Christofides, Mingozzi y Toth

- **Paso 1** $k := 1$
- **Paso 2** Seleccionar un cliente no visitado v_k
- **Paso 3** Para cada cliente no visitado w, calcular δ_{w,v_k}
- **Paso 4** Insertar $w^* = \arg\min_w \delta_{w,v_k}$ en la ruta y aplicar 3-opt
- **Paso 5** Repetir mientras existan clientes factibles. Si no hay más, aumentar k y volver al paso 2

Fase 2: Asignación e inserción de clientes restantes Una vez determinadas las k rutas iniciales, se procede a asignar los clientes restantes. Para cada cliente w no visitado, se determina la ruta r_t en la que su inserción es más económica:

$$t_w = \arg\min_t \delta_{w,v_t} \tag{55}$$

Posteriormente, se calcula una medida de urgencia para cada cliente, basada en la diferencia entre su mejor y segunda mejor opción de inserción:

$$\tau_w = \delta_{w,t'_w} - \delta_{w,t_w} \quad \text{donde} \quad t'_w = \arg\min_{t \neq t_w} \delta_{w,v_t} \tag{56}$$

El cliente con mayor urgencia se inserta primero, y se aplica una mejora local mediante el algoritmo 3-opt.

Fase 2 - Algoritmo de Christofides, Mingozzi y Toth

- **Paso 1** Crear k rutas $r_t = (0, v_t, 0)$ para $t = 1, \ldots, k$
- **Paso 2** Para cada cliente w no visitado, calcular $t_w = \arg\min_t \delta_{w,v_t}$
- **Paso 3** Seleccionar una ruta r_t y calcular, para cada w tal que $t_w = t$, el valor de τ_w
- **Paso 4** Insertar $w^* = \arg\max_w \tau_w$ en la ruta r_t y aplicar 3-opt
- **Paso 5** Repetir mientras haya clientes asociados a r_t. Cuando se vacía, continuar con otra ruta
- **Paso 6** Si aún quedan clientes sin visitar, aplicar nuevamente el algoritmo completo sobre ellos

Este enfoque busca construir rutas balanceadas y de buena calidad mediante una planificación anticipada de la cantidad de rutas y una asignación inteligente basada en urgencias, lo que permite evitar rutas mal formadas hacia el final del proceso. El uso de parámetros como λ permite controlar el peso relativo entre la distancia al depósito y la distancia con el cliente inicial de la ruta.

3.4.3. Asignar Primero, Rutear Después

Los métodos de tipo asignar primero y rutear después (cluster first – route second) se desarrollan en dos etapas. En la primera, se agrupan los clientes en subconjuntos llamados clusters, cada uno de los cuales corresponderá a una ruta en la solución final. Esta fase considera las restricciones de capacidad, de modo que la demanda total de cada cluster no exceda la capacidad del vehículo. En la segunda etapa, se construye una ruta para cada cluster que recorra todos sus clientes. Este problema equivale a un TSP (Problema del Viajante), el cual puede resolverse de forma exacta o aproximada, dependiendo del número de clientes en el cluster[47].

Sweep Heuristic

La heurística de barrido es un método secuencial basado en el enfoque de asignación primero y ruteo después, que organiza a los clientes en grupos antes de generar las rutas de distribución. Este procedimiento consta de dos fases principales:

En la primera fase, los clientes se agrupan en *clusters*, asegurando que la demanda total de cada grupo no supere la capacidad del vehículo asignado. Este agrupamiento facilita la posterior optimización de rutas, ya que permite dividir el problema en subproblemas más manejables.

En la segunda fase, cada cluster generado se transforma en una ruta optimizada, resolviendo un TSP de manera exacta o aproximada según el número de clientes incluidos. De esta forma, se obtiene un conjunto de rutas eficientes dentro de cada grupo previamente definido.

En la heurística de barrido[47], los clusters se forman rotando una semirrecta con origen en el depósito e incorporando clientes a medida que son "barridos"hasta que se alcanza la capacidad máxima permitida. Luego, cada cluster se convierte en una ruta resolviendo un TSP.

Este método se aplica en problemas donde los clientes pueden representarse en un plano cartesiano y las distancias se calculan utilizando la distancia euclídeana. Cada cliente i se define en coordenadas polares (ρ_i, θ_i), con el depósito como origen.

Pasos del Algoritmo:

- **Paso 1 (Inicialización).** Ordenar los clientes por θ en orden creciente. Si dos clientes tienen el mismo θ, priorizar el de menor ρ. Seleccionar un cliente inicial w y definir el primer cluster $C_k = \{w\}$, con $k := 1$.
- **Paso 2 (Selección).** Si todos los clientes están en algún cluster, ir al paso 3. En caso contrario, seleccionar el siguiente cliente w_i. Si w_i puede agregarse a C_k sin violar restricciones de capacidad, incluirlo. Si no, crear un nuevo cluster C_k y asignarle w_i. Repetir hasta clasificar todos los clientes.
- **Paso 3 (Optimización).** Para cada cluster C_k, resolver un TSP con sus clientes.

Por su construcción, las rutas generadas no se superponen, lo que puede ser beneficioso en ciertos escenarios. En la Figura 3.4 se muestra un ejemplo de solución generada por este algoritmo, donde las líneas punteadas representan los límites de los clusters.

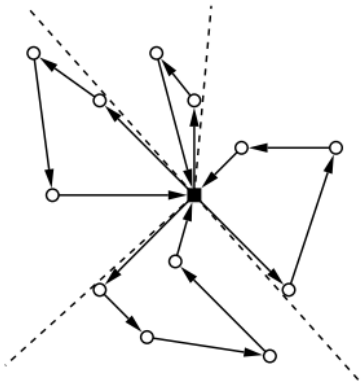

Figura 3.4: Ejemplo de solución obtenida con la Heurística de Barrido[47].

El procedimiento se repite n veces, iniciando cada iteración desde un cliente diferente. Además, se puede aplicar un refinamiento adicional eliminando clientes de rutas ya generadas e insertando aquellos aún no asignados, siempre que esto reduzca el costo total. Los clientes removidos serán considerados en rutas posteriores, permitiendo solapamientos cuando sea necesario.

Fisher y Jaikumar

Fisher y Jaikumar [53] proponen una heurística para resolver el CVRP basada en la descomposición del problema en dos fases: la asignación de clientes a rutas (clustering) y el ruteo dentro de cada clúster. La heurística se apoya en la resolución de un *Problema de Asignación Generalizada (GAP)* para asignar clientes a clústeres de forma eficiente respetando la capacidad de los vehículos.

Fase 1: Selección de semillas Se seleccionan previamente K clientes semilla s_k, con $k = 1, \ldots, K$, sobre los cuales se construirán los K clústeres iniciales. Estos clientes actúan como centros alrededor de los cuales se agruparán los demás clientes.

Fase 2: Resolución del GAP Dado el conjunto de clientes $V \setminus \{0\}$, donde 0 representa el depósito, se busca asignar cada cliente a uno de los clústeres resolviendo el siguiente modelo de optimización:

$$\min_{x_{ik}} \quad \sum_{k=1}^{K} \sum_{i \in V \setminus \{0\}} d_{ik} x_{ik} \tag{57}$$

$$\text{s.a.} \quad \sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i \in V \setminus \{0\} \tag{58}$$

$$\sum_{i \in V \setminus \{0\}} q_i x_{ik} \leq Q \qquad \forall k = 1, \ldots, K \tag{59}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i, k$$

Donde:

- x_{ik} indica si el cliente i es asignado al clúster k.
- q_i es la demanda del cliente i.
- Q es la capacidad máxima del vehículo.
- d_{ik} es el costo de insertar al cliente i en la ruta del clúster k.

El costo d_{ik} se calcula como el incremento de costo al insertar al cliente i en la ruta inicial $(0, s_k, 0)$. Si los costos son simétricos, se simplifica como:

$$d_{ik} = c_{0i} + c_{i,s_k} - c_{0,s_k} \tag{60}$$

Fase 3: Ruteo final Una vez asignados los clientes a sus respectivos clústeres, se resuelve un problema del viajante (TSP) dentro de cada clúster, determinando así el orden óptimo de visitas dentro de cada ruta.

Algoritmo de Fisher y Jaikumar:

- **Paso 1: Inicialización.** Formar K clústeres iniciales seleccionando K clientes semilla s_k.
- **Paso 2: Asignación.** Resolver el GAP para asignar los clientes restantes a cada clúster respetando la capacidad del vehículo.
- **Paso 3: Ruteo.** Resolver un TSP para cada clúster resultante.

Este enfoque permite separar la complejidad del problema de ruteo en una etapa de asignación combinatoria y una etapa de optimización local, mejorando la eficiencia computacional y facilitando la escalabilidad.

Bramel y Simchi-Levi

La heurística propuesta por Bramel y Simchi-Levi [54] comparte la estructura general de la heurística de Fisher y Jaikumar, en la que los clientes se agrupan alrededor de un conjunto de *semillas* o centros de ruta. Sin embargo, en este caso, las semillas no se seleccionan a priori, sino que se determinan resolviendo un *Problema de Localización de Concentradores con Capacidades (CCLP).*

Modelo del Problema de Localización de Concentradores (CCLP) Dado un conjunto de m posibles ubicaciones para instalar concentradores (semillas) con capacidad Q_j para $j = 1, \ldots, m$, y un conjunto de n clientes o terminales con demanda w_i para $i = 1, \ldots, n$, se busca:

- Determinar en qué ubicaciones instalar concentradores.
- Asignar cada cliente a un único concentrador.
- Minimizar la suma de los costos fijos de instalación y los costos de conexión.

La formulación del problema es:

$$\text{mín} \quad \sum_{j=1}^{m} f_j y_j + \sum_{i=1}^{n} \sum_{j=1}^{m} \hat{c}_{ij} x_{ij} \tag{61}$$

$$\text{s.a.} \quad \sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i = 1, \ldots, n \tag{62}$$

$$\sum_{i=1}^{n} w_i x_{ij} \leq Q_j y_j \qquad \forall j = 1, \ldots, m \tag{63}$$

$$x_{ij} \in \{0, 1\}, \quad y_j \in \{0, 1\} \qquad \forall i, j$$

Donde:

- $y_j = 1$ si se instala un concentrador en j, 0 en caso contrario.
- $x_{ij} = 1$ si el cliente i se asigna al concentrador j.
- $\hat{c}_{ij}$ es el costo de conectar el cliente i al concentrador j.

Aplicación al Problema de Ruteo de Vehículos En este contexto, los clientes actúan como terminales y las posibles ubicaciones de concentradores corresponden a posibles semillas. Para cada subconjunto de clientes T_j, se define:

- $f_j = t(T_j)$: el costo de rutear a los clientes en T_j, es decir, el costo de un TSP óptimo.
- $\hat{c}_{ij} = t(T_j \cup \{i\}) - t(T_j)$: el costo incremental de agregar el cliente i al clúster T_j.
- $w_i = q_i$: demanda del cliente i.
- Q_j: capacidad restante del vehículo tras asignar T_j.

Heurística de Localización:

- **Paso 1: Inicialización.** Seleccionar m semillas $T_1, \ldots, T_m$. Para cada j, calcular:

$$f_j = t(T_j), \quad Q_j = Q - \sum_{k \in T_j} q_k \tag{64}$$

Para cada cliente i y semilla j:

$$\hat{c}_{ij} = t(T_j \cup \{i\}) - t(T_j), \quad w_i = q_i \tag{65}$$

- **Paso 2: Localización.** Resolver el CCLP utilizando los valores anteriores.
- **Paso 3: Ruteo.** Resolver un TSP en cada uno de los clusters definidos por los concentradores seleccionados.

Simplificación computacional El cálculo exacto de f_j y $\hat{c}_{ij}$ puede ser costoso. Por eso se propone una simplificación inicial tomando n conjuntos semilla $T_j = \{j\}$, lo que implica:

$$f_j = 2c_{0j} \tag{66}$$

y se proponen dos variantes para calcular $\hat{c}_{ij}$:

- **Seed Tour Heuristic (STH):** Asumiendo simetría, se define:

$$\hat{c}_{ij} = c_{i0} + c_{ij} - c_{j0} \tag{67}$$

  Esta medida estima el costo de insertar el cliente i en la ruta $(0, j, 0)$, como en la heurística de Fisher y Jaikumar.

- **Star Connection Heuristic (SCH):** Se define el costo como:

$$\hat{c}_{ij} = c_{ij} + c_{ji} \tag{68}$$

  lo que modela una ruta en forma de estrella. Esta heurística es simple y, bajo ciertas hipótesis, puede demostrar ser asintóticamente óptima.

3.4.4. Método Rutear Primero, Asignar Despúes

Los métodos denominados *Rutear Primero, Asignar Después* [55] abordan el problema de ruteo de vehículos en dos fases secuenciales. En la primera fase, se construye una ruta que visita a todos los clientes sin tener en cuenta restricciones de capacidad. En la segunda fase, esta ruta se particiona en subrutas factibles que respetan la capacidad de los vehículos.

Fase 1: Construcción de una ruta global Se resuelve un problema de TSP sobre el conjunto de clientes para obtener una ruta global $r = (0, v_1, v_2, \ldots, v_n, 0)$, donde 0 representa el depósito. Esta ruta puede obtenerse de forma exacta o mediante una heurística como 2-opt aplicada sobre una permutación aleatoria inicial.

Fase 2: Partición de la ruta La ruta obtenida se particiona en rutas más cortas, cada una de las cuales debe cumplir con la restricción de capacidad. Este problema puede modelarse como un problema de camino mínimo en un grafo dirigido acíclico $G = (X, E, w)$, donde:

- $X = \{0, v_1, \ldots, v_n\}$ son los nodos.
- Existe un arco $(v_i, v_j) \in E$ si $i < j$ y la demanda total de los clientes $v_{i+1}, \ldots, v_j$ no excede la capacidad del vehículo Q, es decir:

  $\sum_{k=i+1}^{j} d_{v_k} \leq Q$(69)

- El peso del arco representa el costo de realizar una ruta desde el depósito que visite en orden a los clientes entre v_{i+1} y v_j, retornando luego al depósito:

$$w(v_i, v_j) = c_{0,v_{i+1}} + \sum_{k=i+1}^{j-1} c_{v_k,v_{k+1}} + c_{v_j,0} \tag{70}$$

Un arco (v_i, v_j) representa entonces la subruta $(0, v_{i+1}, \ldots, v_j, 0)$. El objetivo es encontrar el camino de menor costo desde el nodo v_0 hasta v_n en el grafo G, lo cual representa la partición óptima de la ruta original en subrutas factibles. Dado que el grafo es acíclico (todos los arcos son de $i < j$), se puede utilizar el algoritmo de Dijkstra o una técnica de programación dinámica eficiente para resolver este subproblema.

Mejoras sobre la ruta inicial Aunque la ruta inicial sea óptima respecto al TSP y la partición se realice de forma exacta, el resultado global no necesariamente será óptimo para

el CVRP. Por ello, es común generar rutas iniciales mediante métodos heurísticos y repetir el procedimiento varias veces con diferentes rutas iniciales. Además, para mejorar la calidad de las subrutas generadas, se puede aplicar una heurística como 2-opt sobre cada subruta individual $(0, v_{i+1}, \ldots, v_j, 0)$, permitiendo reordenar los clientes y reducir la distancia total.

3.4.5. Heuristicas de Mejora

Una vez generada una solución inicial para el problema, se puede aplicar una heurística de mejora con el objetivo de optimizarla. Para ello, se define un conjunto de soluciones $N(s)$, que representan pequeñas modificaciones de la solución actual s. La búsqueda local consiste en reemplazar s por una solución $s^* \in N(s)$ que tenga un menor costo. Este proceso se repite hasta que no sea posible mejorar la solución, alcanzando así un óptimo local[47].

El conjunto de soluciones $N(s)$ se construye aplicando modificaciones específicas a la solución actual, denominadas movidas. En el contexto del VRP, estas pueden clasificarse en dos tipos principales:

- **Movidas dentro de una misma ruta**: Se modifica el orden en que los clientes son visitados, pero sin alterar la asignación de clientes a la ruta.
- **Movidas entre distintas rutas**: Los clientes pueden ser reasignados entre rutas, permitiendo redistribuir la carga y mejorar la eficiencia del recorrido.

Este tipo de estrategias permiten mejorar las soluciones iniciales y explorar configuraciones más eficientes sin necesidad de reconstruir la solución desde cero.

λ-Optimo

Los operadores λ-`opt` son métodos de mejora utilizados en problemas de optimización. El objetivo de estos operadores es mejorar una solución existente de manera local, buscando reducir el costo total de un recorrido.

Definición: Se dice que un recorrido es λ-óptimo (o simplemente λ-opt) si no es posible obtener un recorrido de menor costo al reemplazar cualquiera de sus λ enlaces por otro conjunto de λ enlaces[48].

La cantidad de λ intercambios aplicados a una ruta que atiende a n nodos está dado por la ecuación[56]:

$$2^{(\lambda-1)}(\lambda-1)!\binom{n+1}{\lambda} \tag{71}$$

Donde $\binom{n+1}{\lambda}$ son las posibles formas de eliminar λ arcos de la ruta y $2^{(\lambda-1)}(\lambda-1)!$ es la cantidad de formas que hay de reconectar la ruta.

El operador `2-opt` es una técnica de búsqueda local en la que se intercambian dos aristas del recorrido. La idea básica es que, si un camino tiene cruces o superposiciones, se puede mejorar al "cambiar" o "reemplazar" ciertas partes de la ruta para reducir la distancia total.

Según Shen Lin (1965)[48], una ruta es 2-opt óptima si y solo si no se intersecta a sí misma. Esto implica que no es posible mejorar la ruta intercambiando dos de sus enlaces sin generar una intersección.

Algoritmo `2-opt`:

- Dada un ruta con múltiples nodos.
- Se seleccionan dos pares de nodos en el recorrido que generan una intersección, en este caso (B, E) y (C, D).
- Se intercambian las conexiones de estos nodos, es decir, en lugar de las aristas $B \to E$ y $C \to D$, se sustituyen por $B \to C$ y $D \to E$.
- Si este cambio reduce la longitud total del recorrido, se mantiene la nueva ruta, eliminando la intersección.

Este proceso se repite iterativamente hasta que ya no es posible hacer mejoras.

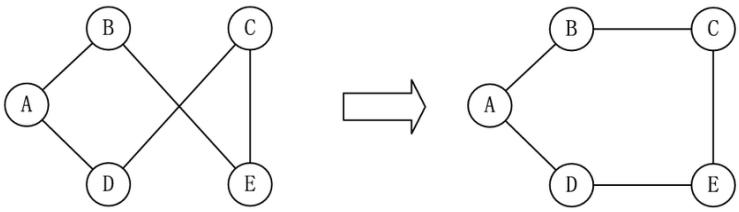

Figura 3.5: Ejemplo del Algoritmo 2-Opt.

El operador `3-opt` es una extensión del `2-opt`. En lugar de intercambiar solo dos aristas, `3-opt` permite realizar intercambios entre tres aristas, lo que potencialmente mejora aún más la ruta.

Algoritmo `3-opt`:

- Al igual que el `2-opt`, partimos de una solución inicial.
- Ahora se seleccionan tres aristas que conectan cuatro nodos, y se prueba varias formas de reorganizar estas cuatro nodos.
- De forma similar al `2-opt`, el objetivo es encontrar una nueva configuración que reduzca el costo total de la ruta.

Existen varias maneras de reorganizar las tres aristas seleccionadas. Hay 7 posibles intercambios diferentes para tres aristas. En general, `3-opt` es más costoso computacionalmente que `2-opt`, pero puede proporcionar mejores resultados en problemas complejos.

Operador Or-opt

El algoritmo **Or-opt** [56], una variante simplificada del método 3-opt, se basa en la reubicación de una secuencia de k clientes consecutivos dentro de la misma ruta. Estos clientes son extraídos de su posición original y recolocados en otro punto de la ruta, manteniendo su orden y adyacencia.

El procedimiento se aplica en etapas: primero con $k = 3$, luego con $k = 2$ y finalmente con $k = 1$, permitiendo así diferentes configuraciones de mejora. La Figura 3.6 ilustra una ruta y las posibles formas de recolocar los tres primeros clientes utilizando esta estrategia.

Dado que cada ruta con n clientes admite $O(n^2)$ posibles modificaciones de este tipo, Or-opt permite explorar diversas alternativas para optimizar la solución sin alterar la secuencia global de clientes visitados.

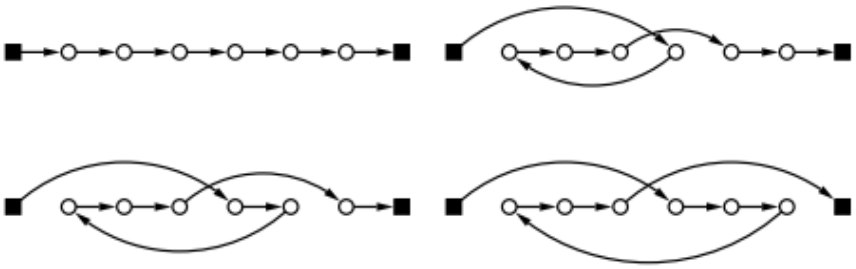

Figura 3.6: Movidas para reubicar los 3 primeros clientes de una ruta. Tomado de [56]

Operadores de Van Breedam

Van Breedam [57] presentó dos operadores diseñados para modificar la asignación de clientes entre diferentes rutas. El primero, denominado **String Relocation**, consiste en extraer una secuencia de m nodos de una ruta y trasladarla a otra, manteniendo el mismo orden en que aparecían originalmente.

El segundo operador, conocido como **String Exchange**, permite intercambiar clientes entre dos rutas. En este caso, una ruta transfiere m clientes a otra y, a cambio, recibe una secuencia de n clientes desde la ruta opuesta.

Para representar estas operaciones, se emplea la notación $(m, 0)$ para una **String Relocation**, lo que indica que m nodos se trasladan sin recibir clientes en retorno. Por otro lado, una **String Exchange** se denota como (m, n), donde m clientes de una ruta son intercambiados por n clientes de la otra.

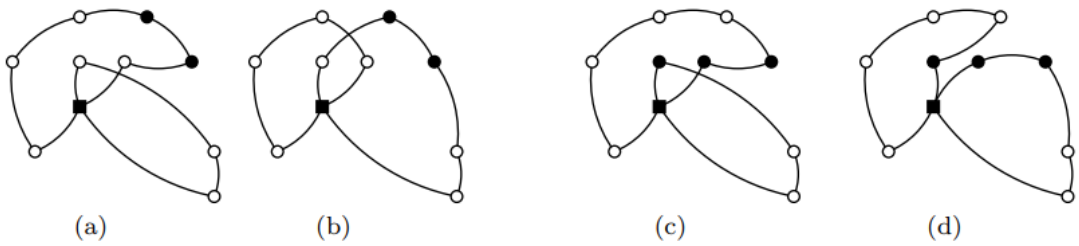

Figura 3.7: Movidas String Realocation y String Exchange. Tomado de [57]

Las Figuras 3.7 (a)-(b) ilustran un ejemplo de **String Relocation (2,0)**, mientras que (c)-(d) se muestra una **String Exchange (2,1)**, donde ambas rutas intercambian clientes.

3.5. Metaheurísticas

Las heurísticas del capítulo anterior proporcionan soluciones eficientes en tiempos de ejecución reducidos, pero su capacidad de exploración del espacio de soluciones es limitada. Para obtener mejores soluciones, es necesario recurrir a técnicas más avanzadas que permitan una exploración más amplia y efectiva.

Las metaheurísticas son estrategias generales para la optimización y búsqueda de soluciones en problemas complejos. Estas técnicas proporcionan un marco flexible que, al adaptarse a un contexto específico, permite desarrollar algoritmos efectivos para problemas difíciles de resolver con métodos tradicionales. Aunque suelen requerir mayor tiempo de ejecución en comparación con las heurísticas clásicas, siguen siendo mucho más eficientes que los enfoques exactos para problemas de gran escala.

Glover las define en 1986 como "Métodos de solución que orquestan una interacción entre procedimientos de mejoramiento local y estrategias de más alto nivel para crear un proceso capaz de escapar de los óptimos locales y realizar una búsqueda robusta del espacio de la solución"[58].

Entre las diversas metaheurísticas existentes, nos centraremos en tres métodos ampliamente utilizados para resolver problemas de ruteo: los Algoritmos de Hormigas, la Búsqueda Tabú y los algoritmos genéticos. Estos métodos se basan en estrategias que permiten explorar el espacio de soluciones de forma más eficaz que las heurísticas clásicas, gracias a mecanismos que combinan aleatoriedad, memoria y adaptación. Estas técnicas buscan mejorar las soluciones iniciales mediante procesos iterativos que aprovechan la cooperación entre agentes o el uso de estructuras de memoria para evitar ciclos. Debido a su capacidad de adaptación y eficiencia, han demostrado ser herramientas efectivas en la resolución de variantes complejas del VRP.

3.5.1. Algoritmos de Hormigas

Los Sistemas de Hormigas[59] están inspirados en el comportamiento colectivo de las colonias de hormigas durante la búsqueda de alimento. Cuando una hormiga encuentra un

camino hacia una fuente de alimento, deposita una sustancia química llamada feromona a lo largo del trayecto. La cantidad de feromona depositada depende de la calidad del alimento y la longitud del camino recorrido.

Si una hormiga no percibe feromonas en su entorno, se desplaza aleatoriamente. Sin embargo, cuando detecta esta sustancia, es más probable que siga los senderos con mayor concentración de feromonas. Esto refuerza aún más dichos caminos, atrayendo a más hormigas en el futuro. Este fenómeno genera un proceso autocatalítico, donde las rutas más transitadas se vuelven progresivamente más atractivas para la colonia, favoreciendo la formación de trayectos óptimos.

Funcionamiento de los Algoritmos de Hormigas

Los Algoritmos de Hormigas replican este comportamiento natural para resolver problemas de optimización combinatoria. Cada hormiga construye una solución basada en dos factores:

1. **Criterio ávido** (greedy), que evalúa qué tan conveniente es tomar una determinada decisión en cada paso.

2. **Memoria colectiva** en forma de feromonas, que refleja la calidad de las decisiones tomadas en iteraciones previas.

El primer problema abordado con esta metodología fue el TSP[59]. En este caso, cada hormiga selecciona su siguiente ciudad a visitar en función de una regla probabilística, que combina la visibilidad de cada nodo y la cantidad de feromona acumulada en cada arco.

La visibilidad de moverse del nodo i al nodo j está dada por:

$$\eta_{ij} = \frac{1}{c_{ij}} \tag{72}$$

donde c_{ij} representa el costo o distancia entre ambos nodos. Además, en la iteración t del algoritmo, cada conexión (i, j) tiene una cantidad de feromona asociada $\tau_{ij}(t)$.

Modelo ANT-Cycle

En el modelo ANT-Cycle, se distribuyen M hormigas en los nodos del problema (generalmente, una hormiga por nodo, es decir, $M = n$). En cada iteración t, cada hormiga construye su solución eligiendo el próximo nodo según una regla probabilística:

$$p_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in \Omega} [\tau_{ih}(t)]^\alpha [\eta_{ih}]^\beta} & \text{si } j \in \Omega \\ 0 & \text{si } j \notin \Omega \end{cases} \tag{73}$$

donde:

- Ω es el conjunto de nodos aún no visitados.
- α y β determinan la influencia relativa de la feromona y la visibilidad en la elección del siguiente nodo.

Actualización de Feromonas

Una vez que todas las hormigas han construido sus soluciones, se actualiza la cantidad de feromona en cada arco según la regla:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \Delta\tau_{ij} \tag{74}$$

donde:

$$\Delta\tau_{ij} = \sum_{k=1}^{M} \Delta\tau_{ij}^k \tag{75}$$

y $\Delta\tau_{ij}^k$ representa la cantidad de feromona depositada por la k-ésima hormiga. Si la solución de la hormiga k tiene longitud L_k y utiliza el arco (i, j), la cantidad de feromona agregada es:

$$\Delta\tau_{ij}^k = \frac{Q}{L_k} \tag{76}$$

donde Q es un parámetro del algoritmo. Si el arco no es parte de la solución de la hormiga, entonces $\Delta\tau_{ij}^k = 0$.

El coeficiente de evaporación de feromona ρ $(0 \leq \rho \leq 1)$ regula la cantidad de feromona que persiste entre iteraciones, evitando la convergencia prematura del algoritmo y favoreciendo la exploración de nuevas soluciones.

Algoritmo Ant System para TSP[47]

1: **Paso 1 (Inicialización).** Colocar una hormiga en cada nodo. Definir $t := 0$ y establecer la cantidad inicial de feromona en cada arco:

$$\tau_{ij}(t) := \tau_0, \quad \forall (i,j) \in E$$

2: **Paso 2 (Construcción).** Para cada hormiga, construir una solución de acuerdo con la regla probabilística dada en la ecuación (51).

3: **Paso 3 (Actualización).** Actualizar la feromona en cada arco de acuerdo con la ecuación (52).

4: **Paso 4 (Terminación).** Incrementar el contador de iteraciones $t := t + 1$. Si $t < T_{\text{máx}}$, colocar una hormiga en cada nodo y regresar al paso 2. En caso contrario, finalizar el algoritmo.

Ant-System con Seleccion Elitista

Una de las modificaciones introducidas al algoritmo *Ant-System* fue la incorporación de una estrategia elitista en la actualización de la feromona [60]. Este enfoque da mayor peso a la mejor solución obtenida en cada iteración, favoreciendo su influencia en la búsqueda de soluciones futuras.

Si la mejor solución encontrada en la iteración t tiene una longitud L^*, la actualización de la feromona para el arco (i,j) se define como:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^* \tag{77}$$

donde:

$$\Delta\tau_{ij}^* = \begin{cases} \sigma \frac{Q}{L^*}, & \text{si } (i,j) \text{ pertenece a la mejor solución} \\ 0, & \text{en caso contrario} \end{cases} \tag{78}$$

Aquí, el parámetro σ representa la cantidad de hormigas elitistas, que simulan haber encontrado la mejor solución y refuerzan la feromona en los arcos que la componen. Este mecanismo mejora la convergencia del algoritmo al intensificar la búsqueda en regiones prometedoras del espacio de soluciones.

Ant-System Híbrido para VRP

La variante *Ant System Híbrido para VRP*, propuesta por Bullnheimer, Hartl y Strauss [61], modifica el algoritmo original *Ant-System* incorporando restricciones de factibilidad en la selección de nodos. En este enfoque, el conjunto de nodos candidatos Ω solo incluye aquellos que pueden insertarse sin violar restricciones. Si no hay nodos disponibles, la hormiga retorna al depósito y comienza una nueva ruta.

El algoritmo emplea selección elitista y aplica un procedimiento *2-opt* tras la construcción de cada solución. La probabilidad de elegir un nodo j se basa en la regla de *Ant-System*, pero con términos adicionales para evaluar el ahorro de Clarke and Wright μ_{ij} y la utilización del vehículo k_{ij}, lo que aumenta el costo computacional.

Para optimizar el rendimiento, los autores redefinen la visibilidad η_{ij} como:

$$\eta_{ij} = c_{i0} + c_{0j} - gc_{ij} + f|c_{i0} - c_{0j}| \tag{79}$$

Esta modificación elimina la necesidad de recalcular las probabilidades en cada iteración, reduciendo el costo computacional. Además, se limita el conjunto de candidatos a los $[n/4]$ vecinos más cercanos y se mantiene la selección elitista con ranking, siguiendo el mismo esquema del algoritmo original.

3.5.2. Búsqueda Tabú

La **Búsqueda Tabú** (Tabu Search) es una metaheurística propuesta por Glover [58] para resolver problemas de optimización combinatoria. A diferencia de las búsquedas locales clásicas, permite aceptar soluciones que empeoran temporalmente el valor de la función objetivo, con el fin de evitar caer en óptimos locales y explorar regiones más amplias del espacio de soluciones.

Durante la iteración t, el algoritmo parte de una solución actual s_t y genera una nueva solución s_{t+1} aplicando una *movida*, es decir, una operación que modifica levemente la solución. La nueva solución s_{t+1} corresponde a la mejor dentro del conjunto de soluciones vecinas $N(s_t)$, que pueden incluir soluciones de peor calidad.

Como existe el riesgo de volver a visitar soluciones anteriores, se emplea una **memoria de corto plazo**, donde se registran ciertos atributos de las soluciones recientes (por ejemplo, los clientes intercambiados entre rutas). Se define un horizonte temporal θ, durante el cual ciertas movidas son consideradas *tabú*, es decir, están prohibidas. A las soluciones generadas por estas movidas se les denomina *soluciones tabú*.

Para evitar que la estrategia tabú restrinja indebidamente el progreso hacia mejores soluciones, se introduce el **criterio de aspiración**, que permite aceptar soluciones tabú si mejoran la mejor solución global obtenida hasta el momento.

Además de estas memorias reactivas, la Búsqueda Tabú puede incorporar mecanismos de **diversificación** y **intensificación**. La diversificación se logra, por ejemplo, penalizando movidas que involucren clientes que ya han sido modificados con frecuencia, forzando así la exploración de nuevas regiones del espacio de soluciones. La intensificación, por otro lado, busca concentrar la búsqueda en regiones prometedoras, por ejemplo, usando vecindades más amplias o aplicando operadores más potentes en soluciones prometedoras.

Búsqueda Tabú aplicada al VRP Existen múltiples variantes del algoritmo de Búsqueda Tabú aplicadas al Problema de Ruteo de Vehículos (VRP), cada una con mecanismos específicos de generación de vecindad, evaluación de movidas y estrategias de control tabú. A continuación se resumen algunas de las más representativas:

- **Algoritmo de Osman** [62]: define vecindades mediante λ-intercambios entre pares de rutas, en donde se transfieren hasta λ clientes entre ellas. Luego de cada movida, se aplica una mejora local (2-opt) en las rutas modificadas. Se utilizan criterios como *Best Admissible* y *First Best Admissible* para seleccionar las movidas.
- **Taburoute (Gendreau, Hertz y Laporte)** [63]: permite explorar soluciones no

factibles respecto a capacidad y distancia, penalizando dichas violaciones en la función objetivo:

$$c'(s) = c(s) + \alpha Q(s) + \beta L(s) \tag{80}$$

donde $Q(s)$ y $L(s)$ son los excesos en capacidad y longitud, respectivamente. Los parámetros α y β se ajustan dinámicamente.

- **Algoritmo de Taillard** [64]: divide el problema en subproblemas (por partición geométrica o árboles de cobertura mínima) y aplica Búsqueda Tabú a cada uno. Cada cierto número de iteraciones, se reoptimiza cada ruta con un algoritmo exacto de TSP (Volgenant-Jonker).
- **Memorias adaptativas (Rochat y Taillard)** [65]: se basa en almacenar rutas frecuentes en buenas soluciones para construir nuevas soluciones combinando esas rutas. Se emplea una heurística de selección probabilística ponderada y un procedimiento de post-optimización por *Set Partitioning*.
- **Algoritmo de Xu y Kelly** [66]: modela la vecindad como una red de flujo en cuatro niveles, permitiendo redistribuir clientes entre rutas. Combina esta vecindad con intercambios clásicos y penaliza la repetición de movidas en los mismos clientes.
- **Ejection Chains (Rego y Roucariol)** [67]: introduce el concepto de *cadenas de expulsión*, en el cual una secuencia de movidas depende de la expulsión de un cliente de una posición para ubicar otro, propagando el efecto en cadena. Se emplea dentro de una Búsqueda Tabú paralela.
- **Algoritmo "Flor"** [68]: organiza la solución como un conjunto de rutas más un camino principal (la flor), sobre el que se aplican cadenas de expulsión para transformarlo. La estructura facilita la exploración de grandes vecindades.
- **Granular Tabu Search (Toth y Vigo)** [69]: reduce el espacio de búsqueda eliminando arcos que conectan clientes lejanos (de bajo potencial). Sólo se consideran arcos por debajo de un umbral dinámico ν, lo cual acelera la ejecución sin perder calidad.

- **Algoritmo DETABA (Barbarosoğlu y Özgür)** [70]: utiliza dos tipos de movidas, TANE y TANEC, que intercambian subconjuntos de clientes entre rutas, guiadas por su cercanía al centroide. Se implementan falsos comienzos y se aplican penalizaciones y 2-opt tras cada movida.

Búsqueda Tabú aplicada al VRPTW

Unified Tabu Search Cordeau, Laporte y Mercier propusieron una variante de Búsqueda Tabú para el *Vehicle Routing Problem with Time Windows* (VRPTW), denominada Unified Tabu Search [71]. Esta metaheurística extiende los principios de Taburoute, permitiendo manejar restricciones múltiples y soluciones no factibles durante la búsqueda.

Se parte de la suposición de que el tamaño de la flota m es conocido y fijo. Las restricciones consideradas incluyen:

- Capacidad de los vehículos.
- Longitud máxima permitida por ruta.
- Ventanas de tiempo para cada cliente.

Durante la búsqueda, se penalizan las violaciones a dichas restricciones incorporando términos en la función objetivo extendida, similar al enfoque adoptado en Taburoute. Sin embargo, en este caso, los coeficientes de penalización se ajustan dinámicamente en cada iteración para adaptarse al comportamiento de la búsqueda.

Las movidas consideradas consisten en seleccionar un cliente, eliminarlo de su ruta actual y reinsertarlo en otra ruta entre dos clientes consecutivos. Al eliminar un cliente de su ruta, los nodos adyacentes se conectan entre sí para mantener la continuidad. Como política tabú, se prohíbe durante θ iteraciones volver a insertar el mismo cliente en la ruta de la que fue eliminado.

Como mecanismo de diversificación, si una movida empeora el valor de la función objetivo, se penaliza su selección utilizando un término adicional ρ_{ir}, que mide cuántas veces el cliente

i fue insertado en la ruta r durante la búsqueda. Esto evita la concentración excesiva de movidas sobre los mismos elementos y promueve la exploración de nuevas configuraciones.

Finalmente, como método de post-optimización, se aplica una versión adaptada del algoritmo GENI (*Generalized Insertion*) específicamente diseñado para problemas con ventanas de tiempo [72], buscando mejorar la mejor solución obtenida al término de la búsqueda tabú.

3.5.3. Algoritmos Genéticos

Los **Algoritmos Genéticos** (AG), introducidos por Holland [73], se inspiran en los principios de la evolución natural y la genética para resolver problemas de optimización y búsqueda. Operan sobre una *población* de soluciones codificadas (individuos), que evolucionan a lo largo de generaciones mediante operadores de selección, cruzamiento y mutación.

Cada individuo representa una solución y es evaluado mediante una *función de fitness* $f(i)$, que mide su calidad relativa. Los individuos con mayor fitness tienen más probabilidades de ser seleccionados para reproducirse. Existen diferentes esquemas de selección, como la *selección proporcional* y la *selección por torneo*.

Operadores evolutivos

- **Cruzamiento:** combina dos individuos padres para generar nuevos hijos. Algunos operadores clásicos en codificaciones binarias son:
  - *SPX* (Single Point Crossover)
  - *DPX* (Double Point Crossover)
  - *UX* (Uniform Crossover)
- **Mutación:** modifica aleatoriamente un individuo para mantener diversidad. En codificaciones binarias puede consistir en invertir bits o permutar elementos.

Aplicación al TSP Para el TSP, las soluciones se representan como permutaciones de nodos. Los operadores clásicos de cruzamiento no garantizan que los hijos sean permutaciones

válidas, por lo que se emplean operadores específicos como el **PMX** (Partially Matched Crossover). Este operador intercambia segmentos y repara duplicados utilizando mapeos entre los elementos.

Aplicación al VRP Existen múltiples adaptaciones de los AG al Problema de Ruteo de Vehículos (VRP), entre las que destacan:

- **GVR (Genetic Vehicle Representation)** [74]: trabaja directamente sobre rutas. Utiliza cruzamiento por inserción de sub-rutas de un padre en rutas del otro, y cuatro operadores de mutación (intercambio, inversión, reubicación de cliente y sub-ruta).
- **Baker y Ayechew** [38]: codifican la asignación de clientes a vehículos. Evalúan fitness aplicando 2-opt y 3-opt sobre cada ruta. Se penalizan las violaciones a las restricciones.

Aplicación al VRPTW Para el VRPTW, además de los anteriores, se han desarrollado algoritmos que integran restricciones de capacidad y ventanas de tiempo. Algunos ejemplos destacados son:

- **GIDEON** [75]: divide el espacio mediante puntos semilla y asigna clientes por sectores. El AG determina la ubicación de las semillas. Se penalizan restricciones y se aplica búsqueda local (-intercambios).
- **GenSAT** [76]: postprocesa las soluciones de GIDEON con Simulated Annealing y Tabu Search, usando vecindades por 2-intercambios.
- **GENEROUS (Potvin y Bengio)** [77]: trabaja directamente sobre soluciones factibles. Propone operadores como SBX (Sequence-Based Crossover) y RBX (Route-Based Crossover), y mutaciones centradas en reducción de vehículos.
- **Berger, Barkaoui y Bräysy** [78]: evolucionan dos poblaciones paralelas (una factible, una con menos rutas), aplicando operadores avanzados y un modelo de estado estacionario.

- **Tan, Lee y Ou** [79]: codifican directamente la agrupación de clientes y aplican cruzamiento sobre agrupaciones. Las rutas se construyen con inserción secuencial y se mejoran con -intercambios.
- **Bräysy y Dullaert** [80]: combinan heurísticas de inserción con un AG basado en operadores I-CROSS y RRC que modifican rutas mediante intercambio de arcos y reinserción de clientes relacionados.
- **Potvin y Dubé** [81]: utilizan un AG para ajustar los parámetros de la heurística de inserción en paralelo. Representan los parámetros como vectores binarios y los evalúan ejecutando la heurística correspondiente.

3.6. Resumen metodos de resolucion VRP

En la tabla 3.4 se presenta a modo de resumen, todos los métodos mencionados en este capitulo con una breve descripción de cada uno.

Método	**Tipo**	**Sección**	**Comentario**
Exacto	Branch & Bound [43]	3.3.1	Divide el problema en subproblemas y calcula cotas inferiores. Así descarta ramas no prometedoras y busca la solución óptima de forma más eficiente.
Exacto	Programación Dinámica [40]	3.3.2	Resuelve el problema dividiéndolo en subproblemas más pequeños y aprovechando la superposición de soluciones parciales. De esta forma construye la solución óptima de manera eficiente en instancias de tamaño reducido.
Heurística	Clark and Wright Savings Heuristic [46]	3.4.1	Combina rutas individuales en una sola ruta más eficiente generando ahorros en distancias; rápido y eficiente.

Método	Tipo	Sección	Comentario
Heurística	Solomon Insertion Heuristic [1]	3.4.2	Construye la solución de manera incremental insertando clientes en rutas existentes según un criterio de costo mínimo.
Heurística	Potvin y Rousseau Insertion Heuristic [50]	3.4.2	Generan múltiples rutas de manera simultánea insertando clientes en paralelo; Útil cuando se cuenta con una flota de más de un solo vehículo.
Heurística	Mole & Jameson Insertion Heuristic [51]	3.4.2	Se basa en el ahorro de Clarke & Wright, pero incorpora reglas de inserción secuencial para mejorar la construcción de las rutas.
Heurística	Christofides, Mingozzi y Toth Insertion Heuristic [52]	3.4.2	Es una heurística de inserción paralela que primero define rutas iniciales y luego las completa mediante reglas de inserción.
Heurística	Sweep Heuristic [47]	3.4.3	Agrupa clientes en clusters según su posición angular, respetando la capacidad de los vehículos. Luego genera las rutas dentro de cada grupo, simplificando la resolución del problema.
Heurística	Fisher y Jaikumar Heuristic [53]	3.4.3	Agrupa clientes en clústeres mediante programación lineal y luego construye rutas para cada grupo.
Heurística	Bramel y Simchi-Levi Heuristic [54]	3.4.3	Extiende el método Sweep incorporando técnicas de descomposición y programación matemática.

Método	Tipo	Sección	Comentario
Heurística	Rutear primero, asignar después [55]	3.4.4	Generan primero una ruta completa sin considerar capacidades y luego la dividen en subrutas que cumplen con las restricciones de los vehículos.
Heurística de mejora	2-opt [48]	3.4.5	Intercambia pares de aristas en la ruta para eliminar cruces y reducir la distancia total; rápido y fácil de implementar, pero puede quedarse atrapado en óptimos locales.
Heurística de mejora	3-opt [48]	3.4.5	Generaliza 2-opt al considerar la eliminación y reconexión de tres aristas; ofrece mejoras mayores sobre la ruta inicial, aunque con mayor coste computacional.
Heurística de mejora	Or-opt [56]	3.4.5	Reubica segmentos de 1 a 3 clientes dentro de la misma ruta para reducir la distancia total; combina eficiencia y flexibilidad, mejorando soluciones iniciales de manera rápida.
Heurística de mejora	Operadores de Van Breedam [57]	3.4.5	Traslada una secuencia de m clientes de una ruta a otra, conservando su orden original para explorar nuevas configuraciones de rutas.
Metaheurística	Algoritmo de Hormigas [59]	3.5.1	Simula el comportamiento de colonias de hormigas mediante feromonas para guiar la construcción de rutas;

Método	**Tipo**	**Sección**	**Comentario**
Metaheurística	Tabu Search [62]	3.5.2	Acepta soluciones que empeoran temporalmente el valor de la función objetivo, con el fin de evitar caer en óptimos locales y explorar regiones más amplias del espacio de soluciones.
Metaheurística	Algoritmos Genéticos [73]	3.5.3	Aplican selección, cruzamiento y mutación sobre una población de soluciones, simulando la evolución natural para encontrar configuraciones óptimas.

Tabla 3.4: Comparativa de métodos exactos, heurísticos, metaheurísticos y heurísticas de mejora.

4. Proyecto DROMBO

4.1. Realidad del Problema

Tacuarembó es el departamento más extenso de Uruguay y presenta una geografía predominantemente rural, con numerosas localidades dispersas y de difícil acceso. Muchas de estas se conectan con la capital departamental a través de rutas de tierra que, en épocas de lluvias intensas, se vuelven intransitables. Esta realidad complica seriamente la conectividad y genera desafíos logísticos considerables, especialmente en lo que respecta al acceso oportuno a servicios y recursos sanitarios.

En este contexto, las policlínicas rurales constituyen el principal punto de atención médica para la población de las zonas rurales. Sin embargo, su correcto funcionamiento depende en gran medida de una logística eficiente que permita el flujo regular de insumos médicos entre ellas y el Hospital de Tacuarembó. Este intercambio se da en ambos sentidos: por un lado, cuando el hospital envía insumos médicos a las policlínicas, y por otro, cuando estas envían al hospital elementos como muestras biológicas: sangre o leche materna. A estos movimientos los denominamos traslados, y se clasifican en dos tipos: *pedidos*, cuando el insumo parte del hospital hacia una policlínica, y *envíos*, cuando el origen es la policlínica y el destino el hospital.

Cada traslado está sujeto a condiciones logísticas específicas. Una de las más relevantes es la urgencia: algunos deben realizarse el mismo día en que se solicitan, mientras que otros admiten mayor flexibilidad y pueden programarse a lo largo de la semana. Además, todos los traslados están asociados a una ventana de tiempo, es decir, un intervalo dentro del cual deben concretarse para garantizar su efectividad tanto en términos clínicos como operativos.

Actualmente, no existe un sistema logístico formalizado para gestionar estos traslados de manera eficiente. En la práctica, la responsabilidad de coordinar estos traslados recae sobre el personal sanitario. Esta situación se agrava particularmente en casos de urgencia, donde la rapidez en la entrega de insumos puede ser crítica para la atención del paciente.

Una de las situaciones que más evidencia estas dificultades es el traslado de leche materna

donada en la que se depende de ambulancias que no siempre están disponibles al momento de la extracción, lo que reduce la vida útil de la leche y limita su aprovechamiento.

La distribución de medicamentos también presenta deficiencias importantes. Actualmente, los envíos se realizan una vez al mes, en base a una estimación del consumo mensual de cada policlínica. Esta metodología, poco precisa, da lugar a dos problemas opuestos: el sobrestock, que genera vencimiento y desperdicio de medicamentos, y el desabastecimiento, que pone en riesgo a los pacientes que necesitan fármacos que no están disponibles cuando los requieren. En muchos casos, esta situación obliga a los usuarios a desplazarse hasta la capital departamental —en algunos casos recorriendo hasta 90 km— para acceder a fármacos, con el consiguiente impacto económico y sanitario.

En respuesta a estos desafíos, el Hospital de Tacuarembó ha incorporado un dron modelo Eiger-03, diseñado específicamente para el transporte de insumos médicos. Este dron tiene una capacidad máxima de carga de 3 kilogramos y un volumen de 15 litros, lo que permite cubrir una parte importante de la demanda, aunque con restricciones en peso y volumen. No obstante, su operación actual es limitada: carece de un sistema automatizado para planificar rutas, gestionar solicitudes o coordinar entregas en función de la prioridad y disponibilidad. Esto impide aprovechar plenamente su potencial y reduce su impacto como herramienta de mejora en la logística sanitaria rural.

La empresa Cielum [82], junto con el apoyo de UNICEF [83] (Fondo de las Naciones Unidas para la Infancia), ANII [84] (Agencia Nacional de Investigación e Innovación) y otros actores, ha iniciado en julio de 2023 un proyecto piloto en Tacuarembó para crear una red de traslados mediante drones. Estos drones, con una autonomía de vuelo de hasta 100 kilómetros, están destinados a conectar el hospital con las policlínicas más alejadas. Este sistema no solo facilita el transporte rápido de insumos, sino que también evita la necesidad de trasladar físicamente a los pacientes o depender de la disponibilidad de ambulancias, lo que reduce los costos asociados y disminuye la huella de carbono al evitar el uso de vehículos terrestres.

En cuanto a la infraestructura, el hospital de Tacuarembó construyó un vertipuerto para

el aterrizaje del dron y cuenta con baterías de repuesto. Inicialmente, el proyecto cuenta con 1 dron y se enfoca en tres policlínicas, con una meta de expansión a diez policlínicas en fases posteriores. Los vuelos son operados por pilotos capacitados por CIELUM, DRONFLIES y RigiTech y son supervisados por la Fuerza Aérea y la DINACIA [85] (Dirección Nacional de Aviación Civil e Infraestructura Aeronáutica), lo que garantiza la seguridad y el cumplimiento de las normativas aéreas.

4.2. Policlínicas

Tacuarembó, cuenta con varias policlínicas distribuidas en sus pueblos rurales, ubicados a distintas distancias de la capital departamental. Estas policlínicas desempeñan un rol vital para la atención de los habitantes de las zonas alejadas, pero enfrentan desafíos significativos debido a la distancia y las dificultades en el acceso, particularmente durante condiciones climáticas adversas.

Las policlínicas rurales, como las de Paso del Cerro, Ansina, Curtina, Piedra Sola, Achar, Tranqueras y Tambores, están a menudo a más de 50 kilómetros del hospital de Tacuarembó, con algunas superando los 100 kilómetros. En particular, en la Figura 4.1 se muestra la policlínica de Piedra Sola y en Figura 4.2 la policlínica de Tambores.

Es importante destacar que, para este proyecto, solo se consideran aquellas policlínicas que cuentan con una ruta directa entre el hospital y la propia policlínica. Por ruta directa se entiende que el dron no debe detenerse en ninguna policlínica intermedia para recargarse antes de llegar a su destino. Por parte del cliente se cuenta con los datos de geolocalización de cada una de las policlínicas en el departamento de Tacuarembó así como el dato asociado a la distancia de las rutas que las conectan.

Figura 4.1: Localidad de Piedra Sola departamento de Tacuarembó.

Figura 4.2: Policlínica de Tambores departamento de Tacuarembó

Estas localidades, con poblaciones reducidas y dispersas, se encuentran en diferentes direcciones tomando como referencia el hospital de Tacuarembó: al norte están Paso del Cerro y Tranqueras; al oeste, Ansina, Toscas de Caraguatá, Paso de los Novillos y Montevideo Chico; al sur, Curtina, Piedra Sola, Achar, Peralta y Sauce de Batoví; y al este, Tambores. En la Figura 4.3 se presenta un mapa del departamento de Tacuarembó que muestra la ubicación de todas estas policlínicas. Actualmente, el dron opera activamente en tres policlínicas: Tambores, Curtina y Ansina. Próximamente se prevé ampliar la cobertura a las localidades de Paso del Cerro y San Gregorio de Polanco, llegando así a 5 policlínicas en el corto plazo, fortaleciendo así la red de traslados médicos en el departamento. En la Figura 4.4 podemos ver un mapa de tacuarembó con estas policlínicas.

Figura 4.3: Realidad topográfica con la ubicación de las diferentes policlínicas rurales.

Una solución eficiente a estos problemas logísticos es crucial para mejorar el acceso a los servicios de salud en estas localidades rurales.

Figura 4.4: Red objetivo para Tacuarembó.

La logística y la eficiencia de los traslados dependen en gran medida de la comunicación verbal y la coordinación directa entre el personal de enfermería de la policlínica y el piloto del dron. No se realizan planificaciones más allá de las necesidades diarias, no se busca optimizar las rutas en función de los traslados y carecen de herramientas automatizadas que puedan mejorar todo este proceso logístico.

4.3. Objetivo del Proyecto

Este proyecto de grado tiene como objetivo la construcción de un software que facilite la planificación de vuelos entre el hospital y las distintas policlínicas, optimizando las rutas de los drones que realizan los traslados. Para ello, se implementará un algoritmo de ruteo enmarcado en la familia de problemas conocidos como Vehicle Routing Problem (VRP), adaptado a las particularidades logísticas y operativas del contexto local.

Los problemas del área del ruteo de vehículos se ajustan de forma natural al contexto

planteado en este proyecto, donde es necesario planificar rutas que permitan atender la demanda de cada policlínica, asegurando que cada una sea visitada una única vez por recorrido. Dentro de esta familia de problemas existen diversas variantes que incorporan restricciones adicionales y permiten modelar situaciones más cercanas a la realidad operativa. Entre ellas se destacan el CVRP (problema de ruteo con capacidad limitada), el VRPTW (con ventanas de tiempo) y el VRPPDTW (con recogida y entrega y ventanas de tiempo), cada uno aportando un mayor grado de precisión para representar condiciones reales como la capacidad de carga del dron, los horarios disponibles en cada policlínica y la necesidad de realizar traslados en ambos sentidos.

Este proyecto busca generar un impacto positivo en el trabajo logístico de traslados entre el hospital y las policlínicas periféricas, optimizando el uso de drones para el transporte de insumos. Se busca reducir el trabajo manual del operador del dron y de los funcionarios de las policlínicas en tareas de coordinación y entrega, estandarizando los procesos por los cuales se gestionan los traslados.

El sistema a construir deberá automatizar la generación de rutas para el operador, de forma que estos respondan eficientemente a las solicitudes de las distintas policlínicas, maximizando la eficiencia operativa.

El desarrollo de esta herramienta mejorará la logística de los traslados de suministros médicos entre el hospital y las policlínicas. Como consecuencia estaremos colaborando de forma positiva en la economía y la calidad de vida en las comunidades rurales de Tacuarembó.

Con el tiempo, se espera que este sistema se expanda a otras regiones del país, promoviendo un modelo de salud más eficiente y accesible.

4.4. La Solución Propuesta

Para alcanzar este objetivo, surge la necesidad de implementar un sistema que optimice la logística de traslados de insumos médicos, con un enfoque en la tecnología y la coordinación centralizada. El proyecto propuesto plantea la creación de un sistema capaz de gestionar

traslados solicitados desde las policlínicas hacia el Hospital de Tacuarembó o viceversa y que sea capaz de crear planes de vuelo diarios de manera automatizada generando rutas de costo mínimo. Definiremos costo mínimo de una ruta mas adelante.

Para abordar el problema de planificación de rutas, se contemplaron distintas formulaciones del problema clásico de ruteo de vehículos (VRP), entre ellas el CVRP (Capacitated VRP), el CVRPTW (VRP con ventanas de tiempo) y el VRPPDTW (VRP con recogida y entrega y ventanas de tiempo). Se analizaron tanto soluciones exactas como aproximadas, considerando siempre las particularidades operativas y logísticas del contexto local.

En base a lo estudiado en el marco teórico del proyecto, se determinó que la mejor alternativa para abordar el problema en cuestión era el uso de heurísticas, dadas sus ventajas en cuanto a simplicidad, velocidad de cómputo y capacidad de adaptación a distintos tamaños de instancia. Dentro de este enfoque, se seleccionó como base teórica la heurística de Insertion propuesta por Solomon (Sección 3.7), ya que ofrece una buena aproximación al problema y contempla un conjunto de restricciones que se ajustan en gran medida a las condiciones reales observadas en Tacuarembó.

La elección de un enfoque secuencial también se justifica por el hecho de que actualmente se opera con un único dron, por lo que no es necesario planificar rutas de manera concurrente. A su vez, esta estructura mantiene la posibilidad de escalar la solución a un mayor número de policlínicas y traslados sin incurrir en altos tiempos de procesamiento.

También se evaluó el uso de metaheurísticas, tales como algoritmos genéticos, búsqueda tabú o algoritmos de colonia de hormigas. Sin embargo, se descartaron debido a que implican una mayor complejidad de diseño, requieren parametrización fina y demandan tiempos de cómputo considerablemente más altos. Dado que el objetivo del sistema es generar rutas viables de forma frecuente y en tiempo reducido, se priorizó la eficiencia práctica por sobre la búsqueda de una solución óptima global, lo cual vuelve más adecuado el uso de heurísticas específicas y simples frente a estrategias de optimización más complejas como las metaheurísticas.

También se decidió no incorporar una heurística de mejora, dado que el objetivo prin-

cipal del sistema no es encontrar una solución óptima, sino una solución factible, rápida y escalable que se pueda ejecutar de forma periódica. Las heurísticas de mejora, si bien pueden refinar soluciones, requieren mayor tiempo de cómputo y complejidad de implementación, lo cual no se justifica en el contexto operativo del sistema.

Características del problema

El costo de una ruta se define a partir de las siguientes variables: la distancia entre la ubicación del dron y la siguiente policlínica, el horario de inicio de servicio en la policlínica (a partir de qué hora es posible atender la llegada del dron) y, por último cuánto tiempo se dispone antes de la finalización del horario de recepción (a partir de que hora no es posible realizar el traslado).

El tiempo de servicio asociado a un traslado en una policlínica está determinado por diversos factores operativos. Entre los más relevantes se encuentran el proceso de carga o descarga del dron a cargo del personal de enfermería, la recarga o sustitución de la batería en caso de ser necesario o otros factores externos que puedan influir. Este tiempo es un parámetro asociado a cada policlínica y se define a partir del análisis histórico de los traslados realizados, utilizando el promedio observado como una estimación representativa del comportamiento en condiciones normales de operación.

Llamaremos ruta al recorrido del dron que parte desde el Hospital pasa por un conjunto de policlínicas donde debe realizar al menos un traslado y termina su recorrido de vuelta en el Hospital.

De esta forma, el operador del dron recibirá las mejores rutas de vuelo para el día en función de la cantidad de pedidos y envíos solicitados por las policlínicas. Esta solución tendrá un gran impacto en la logística del ruteo para el dron, automatizando procesos que de otra forma involucrarían coordinación manual entre el operador y funcionarios de las policlínicas.

Los funcionarios de las policlínicas podrán a través del sistema solicitar traslados (envíos o pedidos) al Hospital. Estos traslados se componen de el usuario solicitante, la policlínica

destino, el tipo de insumo a cargar en el dron, el peso y el volumen del total de insumos solicitados, la ventana de tiempo de atención en la policlínica, la urgencia del traslado y un rango de fechas en las que el traslado puede ser realizado.

Las policlínicas cuentan con compartimentos de tres tamaños distintos, diseñados para estandarizar el empaquetado de los insumos médicos durante el traslado. Esta estandarización permite una representación precisa de la restricción de volumen impuesta por la capacidad del dron. Previamente, se determina la cantidad y el tipo de compartimentos que pueden incorporarse sin exceder el volumen máximo permitido, asegurando así la viabilidad operativa del sistema. Estos tipos de compartimento se califican por tamaños: chico, mediano o grande.

La urgencia de un traslado es una medida que pondera la importancia a nivel sanitario de un traslado. Esta puede tomar los siguientes valores:

- **Alta** – Corresponde a traslados de máxima prioridad, como aquellos requeridos para atender situaciones clínicas urgentes. Deben ser gestionados con inmediatez y tienen preferencia absoluta en la planificación de rutas.
- **Media** – Son traslados que, si bien no requieren atención inmediata, deben resolverse en un plazo razonable. Se espera que sean atendidos en el día o al día siguiente, dependiendo de la disponibilidad del dron.
- **Baja** – Incluye traslados de baja criticidad, que pueden ser programados con mayor flexibilidad. Su resolución puede postergarse varios días siempre que no interfiera con la logística general ni con traslados de mayor prioridad.

La ventana de tiempo de un traslado esta definida por un rango horario donde la policlínica es capaz de recibir el pedido del Hospital y en el caso de los envíos determina el horario en que la policlínica se encuentra disponible para cargar los insumos al dron.

4.5. Requerimientos funcionales y no funcionales

En esta sección se detallan los **requerimientos funcionales** y **no funcionales** que deben ser cumplidos por el sistema para lograr su objetivo de manera eficiente y efectiva.

4.5.1. Requerimientos Funcionales

- **Solicitar Traslado**

  El sistema debe permitir que el personal de las policlínicas y del Hospital realice solicitudes de traslado de insumos médicos entre las policlínicas y el hospital.

  Cada solicitud debe incluir información clave como nombre de la policlínica, tipo de traslado (*ENVIO* o *PEDIDO*), la lista de insumos, urgencia, peso, tipo de compartimento (*CHICO*, *MEDIANO* o *GRANDE*), ventana de tiempo y rango de fechas en las que se puede realizar el traslado.

- **Listar Traslados**

  El usuario debe ser capaz de ver la lista de traslados agendados en el sistema. Cada traslado de la lista debe tener la información sobre sus características (descritas en Solicitar Traslado) y además el estado en el que se encuentra el traslado.

  Los estados posbile de un traslado son:
  PENDIENTE: El sistema recibió los datos del traslado y se encuentra a la espera de ser asignado a una ruta.
  PLANIFICADO: El traslado fue incluido en un ruta, esperando a ser enviada al sistema externo de manejo del dron RigiTech para su posterior uso por parte del piloto.
  CONFIRMADO: La ruta a la cual pertenece el traslado fue confirmada y enviada a Rigitech.
  EN CAMINO: El traslado fue incluido en una ruta y la ruta se encuentra en curso.
  ENTREGADO: El traslado fue recibido/entregado por la policlínica destino.
  RECHAZADO: No fue posible encontrar una ruta factible para entregar el traslado dadas sus restricciones.

- **Planificación de Rutas**

  El sistema debe ser capaz de generar y actualizar rutas optimizadas en función de las solicitudes recibidas, considerando restricciones como capacidad del dron, ventanas de tiempo, tipo de traslado y urgencia.

  Este servicio deberá ser ejecutado de forma recurrente durante el día cada cierto tiempo configurable por el usuario. De tal forma que se tomen en cuenta nuevos pedidos que llegan durante el día.

  Al ser incluidos en una ruta, los traslados son actualizados automáticamente a un nuevo estado *PLANIFICADO*. En caso de que no sea posible generar una ruta factible, los traslados se mantienen en estado *PENDIENTE* esperando ser ruteados mas adelante.

- **Cancelar traslados**

  El sistema ejecuta una rutina una vez al día para identificar y rechazar automáticamente aquellos traslados cuya ventana de tiempo ya ha expirado, es decir, cuando ya no es posible cumplir con el traslado dentro del rango de fechas establecido.

- **Listar Rutas**

  El sistema debe permitir al usuario visualizar el listado de rutas planificadas, incluyendo todos los datos de los traslados asociados y el estado actual de cada ruta, el cual puede ser *PLANIFICADA*, *EN CURSO*, *FINALIZADA* o *CANCELADA*.

- **Consultar Historial de Rutas**

  El sistema debe permitir al usuario acceder al historial de rutas ya finalizadas o canceladas, permitiendo filtrar por fecha o estado. Cada registro debe mostrar los datos relevantes de la ruta y los traslados que la conformaron.

- **Comenzar Ruta**

  El usuario confirma el inicio de una ruta con el objetivo de realizar todos los traslados que esta contiene. Esta acción cambia el estado de los traslados a *EN CAMINO*, marca la ruta correspondiente como *EN CURSO* y genera las operaciones asociadas a cada traslado en la plataforma de RigiTech.

- **Finalizar Ruta**

  El sistema de RigiTech notifica la finalización de una ruta en estado *EN CURSO*. Esta acción actualiza el estado de todos los traslados incluidos en dicha ruta a *ENTREGADO* y marca la ruta como *COMPLETADA*.

4.5.2. Requerimientos No Funcionales

- **Rendimiento**:

  El sistema debe ser capaz de procesar nuevas solicitudes y calcular rutas sin generar demoras significativas, incluso en momentos de alta carga de solicitudes.

- **Escalabilidad**:

  El sistema debe ser escalable para soportar un incremento en el número de policlínicas conectadas, solicitudes simultáneas y capacidad de transporte para la realidad de Tacuarembó.

- **Usabilidad**:

  La interfaz debe ser fácil de aprender y utilizar por parte del personal no técnico, con un diseño orientado a minimizar la carga cognitiva y los errores de usuario.

- **Mantenimiento**:

  El sistema debe ser fácilmente mantenible, con un diseño modular que permita la incorporación de nuevas funcionalidades sin afectar el rendimiento o la estabilidad del sistema.

4.6. Análisis y Modelado

En este capítulo, definiremos el dominio sobre el cual implementaremos nuestra solución, analizando los factores que lo componen, las entidades involucradas, sus características y posibles modelados representativos. Además, exploraremos cómo aplicaremos el modelo del Vehicle Routing Problem (VRP) con sus variantes de capacidad y ventanas de tiempo a la situación de Tacuarembó.

El sistema tiene como base el Hospital de Tacuarembó, que actúa como el depósito general de abastecimiento para el dron, y 13 policlínicas rurales que deben ser abastecidas tanto para recibir como para enviar insumos médicos. Estas policlínicas están distribuidas en diferentes zonas rurales, lo que plantea desafíos logísticos particulares debido a las distancias y la necesidad de optimizar el uso de los recursos disponibles.

En una primera fase del proyecto, se acotará el problema a tres policlínicas, que serán las primeras en poder realizar solicitudes de traslados, permitiendo realizar una prueba piloto y ajustar el sistema antes de su despliegue completo.

Estas policlínicas, cuentan con estaciones de recarga o puntos de recambio de baterías, pudiendo el dron recargar baterías en cualquiera de estos puntos si el piloto lo cree necesario. Estos puntos de recarga permiten extender la autonomía del dron. El tiempo generado por esta variable se contabiliza como unos de los factores del tiempo de servicio en la policlínica en el algoritmo de ruteo.

Para esto se utiliza un parámetro a nivel del sistema donde se establece el promedio del tiempo de servicio en cada policlínica como un valor constante. Con el paso del tiempo y la utilización del sistema se podrá calibrar de manera más adecuada el valor de dicho parámetro en función del histórico de traslados realizados.

El sistema contará con un mecanismo de gestión de solicitudes de traslado para cada policlínica. Este permitirá al personal hospitalario coordinar envíos y solicitar insumos al Hospital. Para asegurar la adopción por parte del usuario final, la interfaz debe ser accesible y de fácil uso, facilitando la gestión de los recursos médicos sin generar una carga adicional para el personal de las policlínicas.

A partir de estas solicitudes, se generará un conjunto de rutas entre el hospital y las policlínicas. Estas rutas deben poder ser recalculadas de manera dinámica durante el día en función de las nuevas solicitudes que ingresen al sistema, una vez comenzada una ruta esta no podrá ser alterada.

Finalmente, podemos definir algunos casos de uso que representan las principales funcionalidades del sistema, permitiendo formalizar el alcance y las capacidades que deberá implementar. Estos casos de uso se detallan en las Tablas 4.5 a 4.8.

La Tabla 4.5 describe el caso de uso *Solicitar Traslado*, en el cual el personal de enfermería de una policlínica puede realizar pedidos o envíos de insumos médicos al hospital.

En la Tabla 4.6 se presenta el caso de uso *Planificación de Rutas*, donde el sistema genera diariamente itinerarios de vuelos optimizados.

La Tabla 4.7 muestra el caso de uso *Listar Rutas*, que permite a los usuarios visualizar las rutas planificadas.

Finalmente, la Tabla 4.8 expone el caso de uso *Comenzar Ruta*, que corresponde a la acción del operador al iniciar un trayecto planificado.

Nombre	Solicitar Traslado
Descripción	Como usuario de enfermería de una policlínica periférica deseo realizar un pedido de uno o más suministros médicos desde el hospital o enviar uno o más suministros hacia el hospital. Los suministros deben ser entregados dentro del plazo estipulado en la solicitud (ventana de tiempo).
Actores	Personal enfermería policlínica
Precondiciones	El suministro médico debe estar disponible en el inventario del hospital. Compartimentos y peso del suministro ajustadas a límites del Dron.
Flujo normal	1. El usuario ingresa al sistema de solicitudes de traslados médicos. 2. El usuario selecciona si el traslado es de tipo *ENVIO* o *PEDIDO*, suministros, tipo de compartimento y urgencia del traslado. 3. El usuario especifica la ventana de tiempo para la entrega del suministro. 4. El usuario hace click en solicitar traslado. 5. El traslado se incorpora al sistema de gestión de traslados.
Poscondiciones	La solicitud del traslado queda registrada en el sistema. Se notifica al usuario si la solicitud se creó correctamente o hubo algún error. En caso de error se indica el error al usuario en pantalla.

Tabla 4.5: Solicitar Traslado

Nombre	Planificación de Rutas
Descripción	Diariamente, el sistema generara las rutas para el día, a partir de los traslados de las policlínicas.
Actores	Usuario, Sistema
Precondiciones	Debe haber al menos un traslado a realizar para el día de hoy.
Flujo normal	1. El sistema separa los traslados en función de la urgencia 2. El sistema construye las rutas a partir de los traslados, priorizando aquellos con mayor urgencia. Para cada solicitud se consideran la policlínica de destino, la ventana de tiempo, la carga y la distancia, a fin de decidir el orden en que deben ser atendidas. 3. El sistema generará una o varias rutas con origen en el Hospital que tendrán como destinos intermedios diferentes policlínicas antes de volver. 4. Se actualiza el estado de los traslados ruteados a *PLANIFICADO*. 5. Se guardarán en el sistema las rutas generadas.
Poscondiciones	El sistema creó un itinerario de vuelos ordenado de costo mínimo.

Tabla 4.6: Planificación de rutas

Nombre	Listar Rutas
Descripción	Como usuario del sistema de gestión de traslados, deseo visualizar el conjunto de rutas planificadas para un día específico, optimizadas según la urgencia de los traslados y las restricciones operativas del dron.
Actores	Usuario
Precondiciones	Debe haberse ejecutado la generación de rutas para la fecha solicitada.
Flujo normal	1. El usuario accede al sistema de gestión de traslados. 2. Selecciona la opción *Ver Rutas*. 3. El sistema recupera las rutas planificadas para los siguientes 5 días a partir de la fecha de hoy. 4. Se muestra el listado de rutas, incluyendo horarios, policlínicas de destino y estado de la ruta.
Poscondiciones	El usuario obtiene la información de las rutas disponibles.

Tabla 4.7: Listar Rutas

Nombre	Comenzar Ruta
Descripción	Como operador del dron, deseo indicar el inicio de una ruta planificada para facilitar el monitoreo del trayecto, mantener la trazabilidad y enviar la información correspondiente a Rigitech, el sistema encargado de ejecutar las operaciónes de vuelo del dron.
Actores	Operador del dron
Precondiciones	La ruta debe estar planificada y en espera de inicio.
Flujo normal	1. El operador selecciona la ruta a iniciar. 2. Confirma el comienzo del trayecto. 3. El sistema marca la ruta como *en curso*. 4. El sistema envía los datos de la ruta a Rigitech para su ejecución.
Poscondiciones	La ruta se encuentra en estado *en curso* y ha sido enviada exitosamente a Rigitech para su ejecución.

Tabla 4.8: Comenzar Ruta

De esta forma y para que tenga un carácter más ilustrativo podemos emular el siguiente diagrama que represente estas diferentes entidades de dominio y brinda el contexto en el cual interactúan unas con otras.

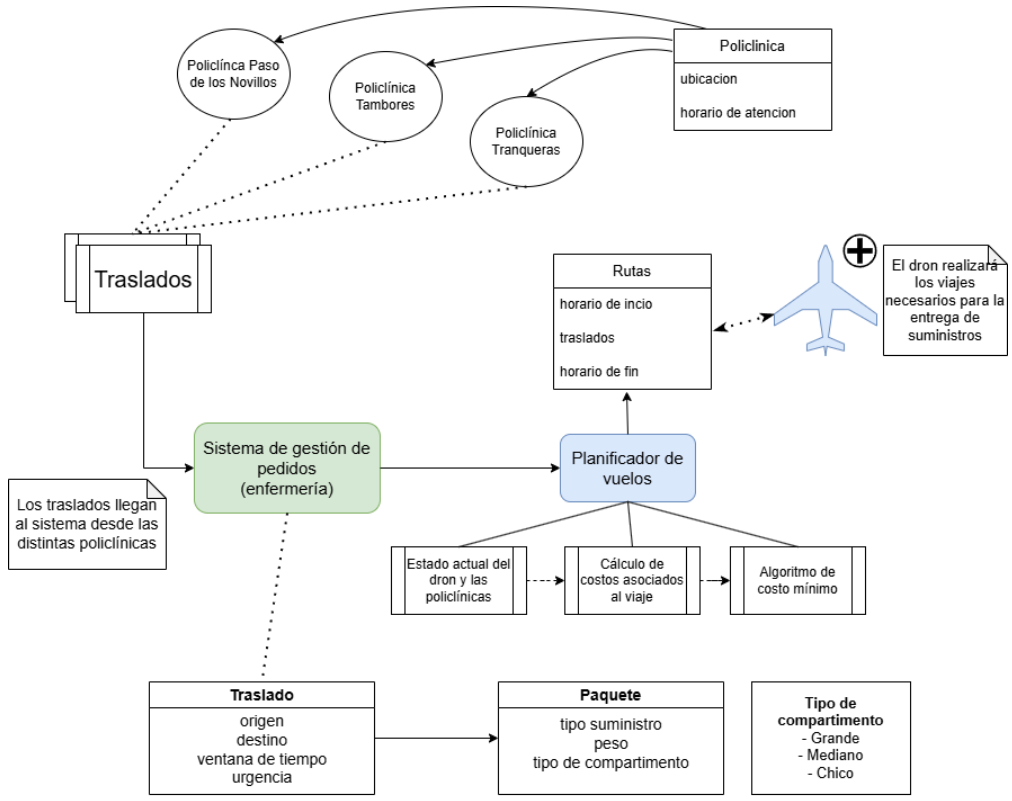

Figura 4.5: Diagrama de alto nivel de entidades del sistema.

El diagrama de la Figura 4.5 muestra cómo los distintos módulos del sistema interactúan para planificar y ejecutar los traslados médicos. Las solicitudes ingresadas desde las policlínicas llegan al sistema de gestión de pedidos, que las centraliza y las envía al planificador de vuelos. Este componente consulta el estado del dron y las policlínicas, calcula los costos y genera las rutas, agrupando traslados según criterios como urgencia y disponibilidad. Final-

mente, las rutas se ejecutan mediante el dron, cerrando el ciclo con la entrega de los insumos y la actualización del sistema.

5. Implementación

Para generar las rutas en función de los traslados de las policlínicas, se evaluó el método más adecuado considerando tanto métodos exactos como heurísticas. Dado el contexto del problema, donde la eficiencia computacional y la escalabilidad son factores clave, se determinó que la heurística de *Insertion Heuristic* es la más adecuada por distintos factores:

- Las restricciones que esta heurística presenta, permiten un buen modelado de la realidad del problema.
- Una heurística nos permite una mejor escalabilidad en función de la cantidad de policlínicas y traslados. Por lo que el algoritmo se adaptará mejor a mayor volumen de procesamiento.

El algoritmo a implementar construye la ruta de manera secuencial, seleccionando en cada iteración el mejor pedido para añadir en la posición óptima dentro de la ruta.

El uso de una heurística en lugar de un método exacto se justifica por la naturaleza combinatoria del problema. Dado que el ruteo de vehículos con ventanas de tiempo y capacidad (CVRPTW) es un problema NP-duro, los métodos exactos como Branch & Bound pueden volverse computacionalmente inviables a medida que el número de policlínicas y traslados aumenta. La heurística *Insertion Heuristic* ofrece una alternativa eficiente que permite encontrar soluciones factibles en tiempos aceptables, sin comprometer la calidad de las rutas generadas. Asimismo, su capacidad de adaptación a variaciones en la demanda y restricciones operativas lo convierte en una opción robusta para la planificación logística del dron.

En su tesis, Røpke[30] investiga tanto algoritmos heurísticos como exactos para abordar diversas variantes del VRP. Los resultados demuestran que las heurística ofrecen soluciones de alta calidad en tiempos computacionales significativamente menores en comparación con los métodos exactos, especialmente en instancias de gran tamaño.

5.1. Algoritmo de ruteo

El algoritmo de *Insertion Heuristic* comienza seleccionando todos los traslados pendientes de ser entregados, incluyendo tanto envíos como pedidos, cada uno identificado por su tipo correspondiente. Luego, estos traslados se organizan en colas según su nivel de prioridad, de manera que se procesen primero aquellos con mayor urgencia. El proceso de planificación se divide en 3 instancias de ruteo, con el objetivo de construir secuencialmente las rutas en función de las prioridades establecidas. El algoritmo 4 rutea entonces la mayor cantidad de traslados para un día, siendo un día el intervalo de tiempo definido por la ventana de tiempo del hospital. Este algoritmo se repite para cada día de la semana.

Algorithm 4 Planificación de Rutas Diarias con Priorización

Parámetros: *traslados_prioridad_alta*, *traslados_prioridad_normales*, *traslados_prioridad_baja*, *matriz_distancias*, *policlinicas*

Salida: Retorna una lista de rutas y una lista de traslados que no pudieron ser ruteados.

1: *rutas* ← lista vacía
2: *traslados_no_ruteados* ← lista vacía
3: *traslados_prioridad_alta* ← lista de traslados de prioridad alta sin rutear.
4: *traslados_prioridad_normal* ← lista de traslados de prioridad normal sin rutear.
5: *traslados_priorida_baja* ← lista de traslados de prioridad baja sin rutear.
6: **while** Existe al menos 1 traslado posible de rutear y la ventana de tiempo del hospital aún no terminó **do**
7: *ruta*, *traslados_no_ruteados* ← ResolverRuteo(*traslados_prioridad_alta*, *ruta*, *matriz_distancias*, *policlinicas*, *traslados_no_ruteados*)
8: *ruta*, *traslados_no_ruteados* ← ResolverRuteo(*traslados_prioridad_normales*, *ruta*, *matriz_distancias*, *policlinicas*, *traslados_no_ruteados*)
9: *ruta*, *traslados_no_ruteados* ← ResolverRuteo(*traslados_priorida_baja*, *ruta*, *matriz_distancias*, *policlinicas*, *traslados_no_ruteados*)
10: **if** ruta con traslados asignados **then**
11: *rutas* ← *ruta*
12: **end if**
13: **end while**
14: **return** *rutas*, *traslados_no_ruteados*

El algoritmo Resolver Ruteo (algoritmo 5) recibe como entrada la cola correspondiente a los traslados pendientes y la ruta generada hasta ese momento. En la primera ejecución, cuando se procesan los traslados de mayor urgencia, no existe una ruta previa, por lo que el algoritmo inicia la construcción desde cero, creando la ruta base e incluyendo la mayor cantidad posible de traslados urgentes. En las siguientes ejecuciones, correspondientes a traslados de menor urgencia, el proceso reutiliza y extiende la ruta ya construida, insertando traslados adicionales siempre que no se violen las restricciones operativas.

El algoritmo recorre los traslados aún no ruteados, en primera instancia se verifica la restricción de capacidad y volumen (peso y compartimento). Para los traslados de tipo *pedido*, desde el hospital hacia la policlínica, se asegura que la carga total al salir del hospital, el dron no exceda su capacidad máxima al agregar el traslado. Para los traslados de tipo *envío*, recolectados en la policlínica y transportados al hospital, se verifica que el dron tenga espacio disponible al momento de la recolección en la policlínica y mantenga este espacio reservado hasta su vuelta al Hospital. Una vez filtrados los traslados factibles, el algoritmo determina el menor costo de inserción para cada uno dentro de la ruta existente. Definimos el costo mínimo de inserción en función de la siguiente ecuación

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) + \alpha_3 c_{13}(i,u,j) \tag{81}$$

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \lambda d_{ij} \tag{82}$$

$$c_{12}(i,u,j) = b'_j - b_j \tag{83}$$

$$c_{13}(i,u,j) = l_u - b_u \tag{84}$$

La función que queremos minimizar esta dada por c_1, la cual representa el costo total de insertar el traslado u en la ruta, ubicándolo después del traslado i y antes del traslado j.

La función c_{11} calcula el costo de insertar el traslado u entre i y j en función de la distancia adicional que el dron debe recorrer para llegar a u. En otras palabras, se prioriza insertar el traslado más cercano a la ruta. Esta función busca optimizar la distancia total recorrida por el dron en una ruta.

Por otro lado, la función c_{12} evalúa el impacto de insertar el traslado u entre los traslados i y j en el horario de inicio del servicio de j; es decir, mide cuánto se retrasa la atención del traslado j al incorporar u en la ruta.

Para calcular los valores de b'_j y b_j, se consideran los tiempos de servicio específicos de cada policlínica. Estos tiempos se calibran utilizando datos del sistema y el historial de pedidos, con el fin de obtener un promedio representativo adaptado a la realidad de cada centro. Como se mencionó anteriormente, los tiempos de servicio pueden verse afectados por diversos factores variables, tales como la recarga o el recambio de baterías, la disponibilidad del personal de enfermería, tiempo de carga y descarga del dron, entre otros.

Finalmente, c_{13} evalúa el costo de insertar el traslado u en función del tiempo restante hasta el cierre de su ventana de atención. De esta forma, se priorizan aquellos traslados cuya ventana está más próxima a finalizar, lo que permite optimizar el uso del tiempo disponible y minimizar el riesgo de no cumplir con las restricciones horarias. Esta estrategia favorece la asignación de traslados con menor flexibilidad temporal, mejorando la eficiencia general del ruteo.

El algoritmo cuenta con parámetros α_1, α_2, α_3 de ajuste que sirven para ponderar la importancia de cada uno de los costos anteriores en el costo total. De esta forma el usuario puede configurar con el comportamiento del algortimo según crea conveniente en función de cada costo. Estos parámetros cumplen las siguentes restricciones $\alpha_1, \alpha_2, \alpha_3 \geq 0$ y $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

En nuestro caso, utilizamos el mismo valor $1/3$ para los tres parámetros, dando la misma importancia a cada una de ellas en el algoritmo.

El algoritmo cuenta con algunos recursos en memoria que ayudan a optimizar los tiempos de cómputo. Por ejemplo, se almacena una matriz de costos asociados a la distancia que el dron debe recorrer entre el Hospital de Tacuarembó y las policlínicas, o entre policlínicas. Esta matriz contiene las distancias se corresponden a rutas existentes que el dron utilizará durante la operación. La matriz es precalculada y cargada en memoria para ser reutilizada durante el proceso de planificación.

Se guardan en memoria cada uno de los tiempos de inicio de servicio y capacidad disponible en cada traslado una vez es agregado a la ruta. Estos a su vez se actualizan si corresponde en función de los nuevos traslados que se agregan. Esta decisión nos permite ahorrar un gran volumen de operaciones necesarias en cada iteración del algoritmo.

Al seleccionar un traslado candidato para insertarlo en una ruta, se verifica que la ruta conserve su factibilidad. Para ello, se calcula el *Push Forward* asociado a la inserción, que representa el desplazamiento del inicio de servicio de cada traslado posterior al insertado, y se comprueba que no se incumpla ninguna ventana de tiempo.

Además, se valida que la carga (peso y volúmen) acumulada del dron, tanto por pedidos como por envíos, no afecte la factibilidad de los traslados ya existentes en la ruta.

Adicionalmente, se realiza un control específico para traslados de tipo *ENVIO*. En este caso, al insertar un nuevo traslado, se verifica que todos los traslados posteriores cuenten con suficiente capacidad para realizar la recolección de los paquetes que ya han sido asignados. De este modo, se garantiza que la inserción de un traslado no provoque conflictos de capacidad en las operaciones de recolección ya planificadas.

A continuación, se presenta el pseudocódigo correspondiente a la implementación del algoritmo. Siendo *traslados_a_rutear* los traslados sin una ruta asignada y *ruta_existente* la ruta que se creó hasta el momento en función de la prioridad de los traslados:

Algorithm 5 Resolver Ruteo con inserción Heurística

Parámetros: Lista de traslados no asignados *traslados_a_rutear*, ruta predefinida *ruta_existente*, Matriz de distancias entre policlínicas *matriz_distancias*, lista de policlínicas del sistema *policlinicas*, lista de traslados que no se pudieron rutear para el día de hoy *traslados_no_ruteados*.

Salida: Ruta óptima y traslados que no pudieron ser ruteados en el día.

1: **if** existe $ruta\_predefinida$ **then**
2: Calcular tiempo de inicio de servicio para los traslados de la ruta
3: Calcular el peso y los compartimentos utilizados en la ruta
4: **else**
5: Inicializar una ruta con origen en el Hospital
6: **end if**
7: **while** True **do**
8: $costo\_minimo \leftarrow \infty$
9: $mejor\_traslado \leftarrow$ `None`
10: $mejor\_posicion\_en\_la\_ruta \leftarrow$ `None`
11: **for all** traslados no ruteados **do**
12: Calcular el costo de inserción c_1 y la mejor posición
13: Verificar restricciones de tiempo, carga y Push Forward
14: **if** el costo es menor que *costo_minimo* **then**
15: Actualizar *mejor_traslado*, *costo_minimo* y *mejor_posicion_en_la_ruta*
16: **end if**
17: **end for**
18: **if** *mejor_traslado* es `None` **then**
19: Terminar la construcción de la ruta agregando al Hospital como último traslado
20: **break**
21: **else**
22: Insertar *mejor_traslado* en la mejor posición encontrada
23: Actualizar la carga del dron, los tiempos de inicio de servicio de los traslados de la ruta en función del push forward.
24: Remover el traslado de la lista de traslados no ruteados.
25: **end if**
26: **end while**
return ruta óptima y traslados no ruteados

Luego del procesamiento, el algoritmo devuelve como resultado final en formato JSON una lista que contiene las rutas que se realizarán ese día. Cada elemento de esta lista representa una ruta con origen y destino en el Hospital. Dichas rutas a su vez contienen una lista con los traslados en el orden que deben ser atendidos, cada uno con el tiempo en el que el dron llegará para atender el traslado. El algoritmo también retornará aquellos traslados que no pudieron ser agregados a ninguna ruta dadas las restricciones existentes. De esta forma se define la estructura de la respuesta en formato JSON por el algoritmo 4 (utilizando las soluciones parciales del algoritmo 5) de la siguiente manera:

```
conjunto_rutas_del_dia = [
 [
    {
      "traslado_id": "ad5393f9-1443-4905-b188-552a45370f67",
      "posicion_en_la_ruta": "1",
      "hora_de_llegada_estimada": "12:45",
    },
    {
      "traslado_id": "98cb346a-1438-4829-9846-af4007126d6a",
      "posicion_en_la_ruta": "2",
      "hora_de_llegada_estimada": "13:45",
    }
    ...
 ],
 ...
]

no_ruteados = ["51be14b6-f7ec-4379-9147-c3e46df15693", "566a3f5c-d59d-4225-b61d-2e0920b6d110"]
```

No se ilustran en el ejemplo de arriba los objetos que simulan el hospital (salida/llegada al depósito) que se modelan en todas las rutas generadas con sus respectivas horas de llegada.

Este formato se eligió por su simplicidad y compatibilidad con las estructuras nativas de Python, permitiendo organizar de forma clara los resultados y facilitando su uso en etapas posteriores del procesamiento dentro del backend.

5.2. Arquitectura

Se cuenta con una arquitectura sencilla y modular compuesta por dos nodos principales y una base de datos centralizada, como se ilustra en la Figura 5.1. Uno de los nodos es la aplicación web, que constituye la interfaz de interacción para los distintos perfiles de usuario. A través de estas aplicaciones, tanto el personal de enfermería de las policlínicas como el piloto del dron acceden a las funcionalidades del sistema. Asimismo, existe una dependencia externa con la aplicación de RigiTech, a la cual se envía la información de las rutas calculadas para crear las operaciones que posteriormente ejecuta el piloto del dron.

La aplicación web se separa en dos componentes: la interfaz para las policlinicas y la interfaz para el piloto del dron. Estas cuentan con diferentes url de acceso para cada caso. La interfaz para policlínicas está diseñada para permitir a los centros de salud solicitar traslados y consultar el estado de los mismos. Por su parte, la interfaz web del piloto está orientada a brindar al operador del dron la información necesaria para revisar los traslados y las rutas generadas y en caso de que se desee comenzar alguna de las mismas, enviar la información de rutas planificadas a la plataforma de RigiTech para crear las operaciones correspondientes y ejecutarlas.

El segundo nodo está conformado por el backend del sistema, que incluye una API central encargada de recibir y procesar las solicitudes provenientes de las aplicaciones web. Esta API interactúa con una capa de servicio, donde se implementa la lógica de negocio del sistema, y con un planificador de rutas que funciona como una tarea recurrente. El planificador analiza los traslados pendientes y genera rutas optimizadas, las cuales son almacenadas y consultadas desde la base de datos.

Toda la información del sistema se persiste en una base de datos compartida, a la cual acceden tanto la capa de servicio como el planificador. Esta estructura permite una operación eficiente y una clara separación de responsabilidades.

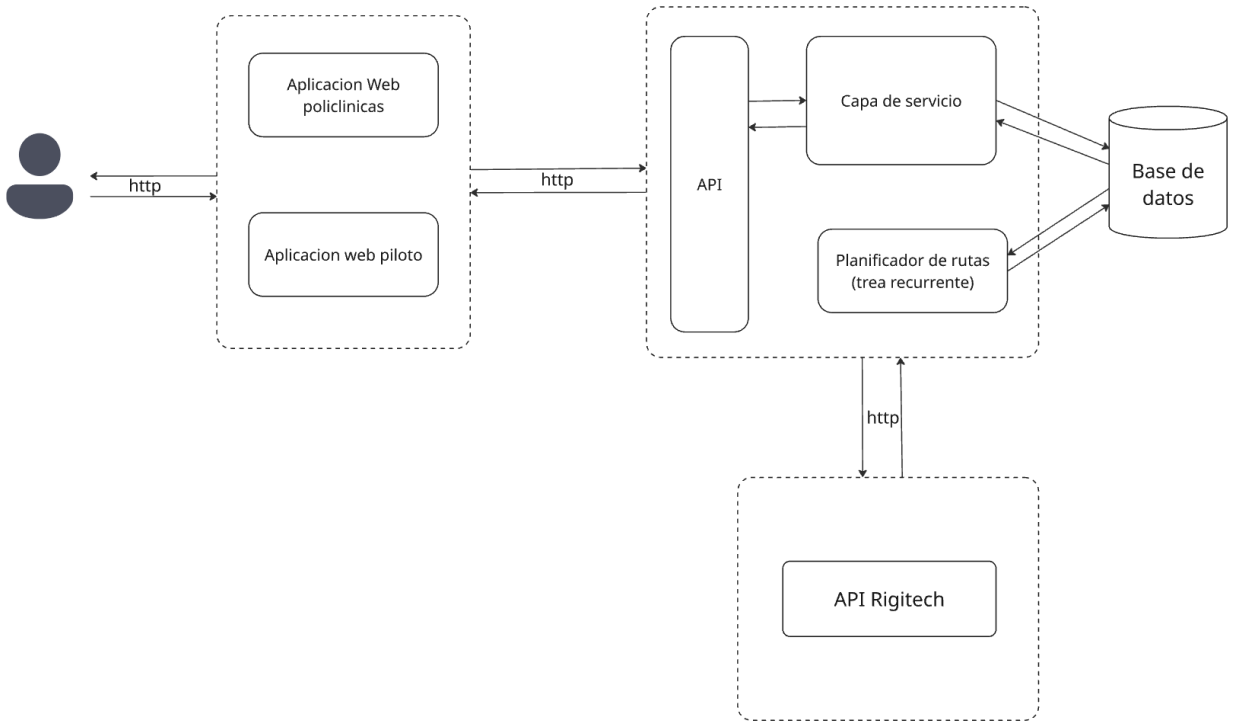

Figura 5.1: Diagrama de la arquitectura del sistema.

5.3. Aplicación web

Para el desarrollo de la aplicación web se utiliza JavaScript[86] junto con la biblioteca ReactJS[87]. La elección de esta tecnología se fundamenta en su alta adopción a nivel mundial para el desarrollo de aplicaciones web modernas y en el amplio conocimiento del equipo en su uso. ReactJS permite la creación de interfaces dinámicas y eficientes gracias a su arquitectura basada en componentes reutilizables y su capacidad de renderizado eficiente mediante el Virtual DOM.

Para la estilización de la interfaz de usuario, se emplean dos bibliotecas ampliamente utilizadas en la comunidad de desarrollo web:

- **Material-UI**: Un conjunto de componentes pre-diseñados basados en las directrices de diseño de Google Material Design[88], lo que proporciona una apariencia moderna y estandarizada a la aplicación.

- **Tailwind CSS**[89]: Un framework de utilidad para CSS que permite definir estilos directamente en los elementos mediante clases predefinidas. Integrado con React, facilita el diseño modular y reutilizable de componentes sin necesidad de archivos de estilo separados.

La comunicación entre el frontend y el backend se realiza a través de una API REST proporcionada por el backend, utilizando el formato JSON para la transferencia de datos. Este enfoque garantiza una integración eficiente y estructurada, permitiendo que el frontend pueda consumir y actualizar información de manera fluida y en tiempo real.

Además, se han implementado buenas prácticas de desarrollo frontend, tales como la gestión del estado global mediante herramientas como *React Context*[90], el manejo eficiente de peticiones HTTP mediante *Axios*[91] y la implementación de rutas dinámicas con *React Router*[92], lo que mejora la navegación y la experiencia de usuario.

5.4. Backend

Para la gestión del backend del sistema, se diseñó e implementó una aplicación en Python[93] que expone una API web a través del framework Flask[94]. La elección de esta tecnología se fundamentó en varios aspectos clave: su simplicidad en el desarrollo y la baja curva de aprendizaje. Además, el equipo posee un amplio dominio en Python, lo que facilitó la integración de la solución de manera ágil y eficiente.

La API web proporciona una serie de servicios diseñados para gestionar los traslados de las policlínicas y la planificación de rutas del dron. Entre los principales servicios disponibles, se incluyen:

- **Crear traslado**: Permite que las policlínicas generen solicitudes de insumos al Hospital, proporcionando detalles como el tipo de insumo, la cantidad y la ventana de tiempo deseada para la entrega.
- **Listar rutas**: Dado un día específico, devuelve el conjunto de rutas planificadas, optimizadas en función de los traslados registrados y las restricciones operativas del dron.

- **Listar traslados**: Permite recuperar la lista completa de traslados realizados, filtrando por estado, fecha o nivel de urgencia.
- **Borrar traslado**: Proporciona la funcionalidad para eliminar un pedido que aún no haya sido procesado en la planificación de rutas.
- **Comenzar ruta**: Permite que el operador del dron indique el inicio de una ruta programada, facilitando el monitoreo del trayecto en tiempo real.

Para garantizar la eficiencia operativa, el sistema incorpora un proceso de planificación automatizado que se ejecuta diariamente para generar las rutas del dron en función de los traslados pendientes. Este proceso es gestionado mediante la biblioteca *APScheduler* [95], la cual permite programar la ejecución del cálculo de rutas de manera recurrente a una hora predefinida. La automatización de este cálculo minimiza la intervención manual y asegura que las rutas sean generadas de manera óptima, considerando aspectos como:

- La priorización de traslados según su nivel de urgencia.
- La optimización del uso del dron, minimizando tiempos de espera y distancia recorrida.
- El cumplimiento de restricciones operativas, como la capacidad de carga del dron y las ventanas de tiempo definidas por el hospital y las policlínicas.

Por último, el backend se integra con la API REST de RigiTech para crear las operaciones correspondientes a las rutas calculadas en el momento de su inicio. Esta integración permite registrar cada traslado de la ruta como una operación independiente dentro de la plataforma de RigiTech, desde donde luego será gestionada por el piloto durante la ejecución del vuelo.

La información se transmite mediante solicitudes HTTP tipo POST con un cuerpo en formato JSON. Por cada traslado incluido en una ruta planificada, se genera un payload con la información necesaria para crear una operación en RigiTech. El siguiente ejemplo muestra la estructura del JSON enviado:

```
{
    "project": 51,
```

```
    "name": "H. de Tacuarembó a Piedra Sola",
    "scheduledTime": "10:00",
    "route": 1360,
    "payload": 2500,
    "drone": 10,
    "batteries": ["CIE-ER-02-042"],
    "pic": 30000,
    "groundOperators": [30001],
    "comments": "Traslado Amoxicilina y Omeprasol a Piedra Sola"
}
```

En este ejemplo:

- `project`: identifica el proyecto dentro del sistema RigiTech.
- `name`: es una descripción legible del tramo del traslado.
- `scheduledTime`: indica la hora estimada de inicio.
- `route`: refiere al identificador de la ruta en RigiTech.
- `payload`: representa el peso en gramos del paquete a trasladar.
- `drone`, `batteries`, `pic` (pilot in command) y `groundOperators`: identifican los recursos involucrados en la operación.
- `comments`: permite registrar observaciones sobre el contenido del traslado.

Por cada traslado se crea una operación individual. Además, se genera la operación adicional para el retorno al Hospital de Tacuarembó.

La API de RigiTech permite, entre otras funciones, registrar nuevas operaciones, consultar el estado de vuelos programados o completados o asignar recursos como drones, baterías u operadores a cada operación. Esta integración automatiza el proceso de creación de vuelos y asegura una transición fluida entre la planificación de rutas y la ejecución operativa por parte del piloto. Generando una agenda diaria para el dron la cual es posible visualizar en la interfaz web de RigiTech.

Adicionalmente, la arquitectura del backend ha sido diseñada para ser escalable y adaptable a futuras mejoras, permitiendo la integración de nuevas funcionalidades y la optimización continua del sistema. Gracias a estas características, el backend proporciona una solución robusta y flexible que facilita la gestión eficiente de los traslados y la planificación de rutas de entrega mediante drones.

5.5. Base de datos

El sistema cuenta con un modelo de base de datos relacional implementado en PostgreSQL 15[96], como se muestra en la Figura 5.2. Se trata de una herramienta de software libre conocida por su robustez, flexibilidad y facilidad de uso. Esta base de datos permite modelar de forma eficiente las relaciones entre entidades clave como traslados, rutas, policlínicas y operaciones, y fue elegida por su capacidad para integrarse fácilmente con sistemas web modernos y por el soporte activo de su comunidad de desarrollo. PostgreSQL está disponible bajo una licencia permisiva y es ampliamente utilizada tanto en entornos académicos como industriales.

La estructura se compone de varias entidades que representan los elementos fundamentales del dominio, así como sus relaciones. La entidad **Policlínica** representa los centros de salud desde los cuales se solicitan los traslados. Cada policlínica posee una identificación única, nombre, coordenadas geográficas, un tiempo promedio de espera (tiempo de servicio) y su horario de atención. Una policlínica puede estar relacionada con múltiples traslados.

La entidad **Traslado** es el núcleo del sistema. Representa una solicitud concreta de transporte de suministros médicos, con atributos como tipo de traslado, fecha de solicitud, fecha de inicio y fin para realizar el traslado, solicitante, horario de atención, tipo de compartimento, peso, nivel de urgencia y estado. Un traslado puede involucrar uno o más suministros.

La entidad **Suministro** representa los insumos médicos a transportar. Incluye un identificador, nombre, cantidad, peso e indicaciones especiales.

La entidad **Ruta** representa un recorrido planificado que puede incluir múltiples traslados. Cada ruta está vinculada a una o varias operaciones, e incluye su identificador, fecha,

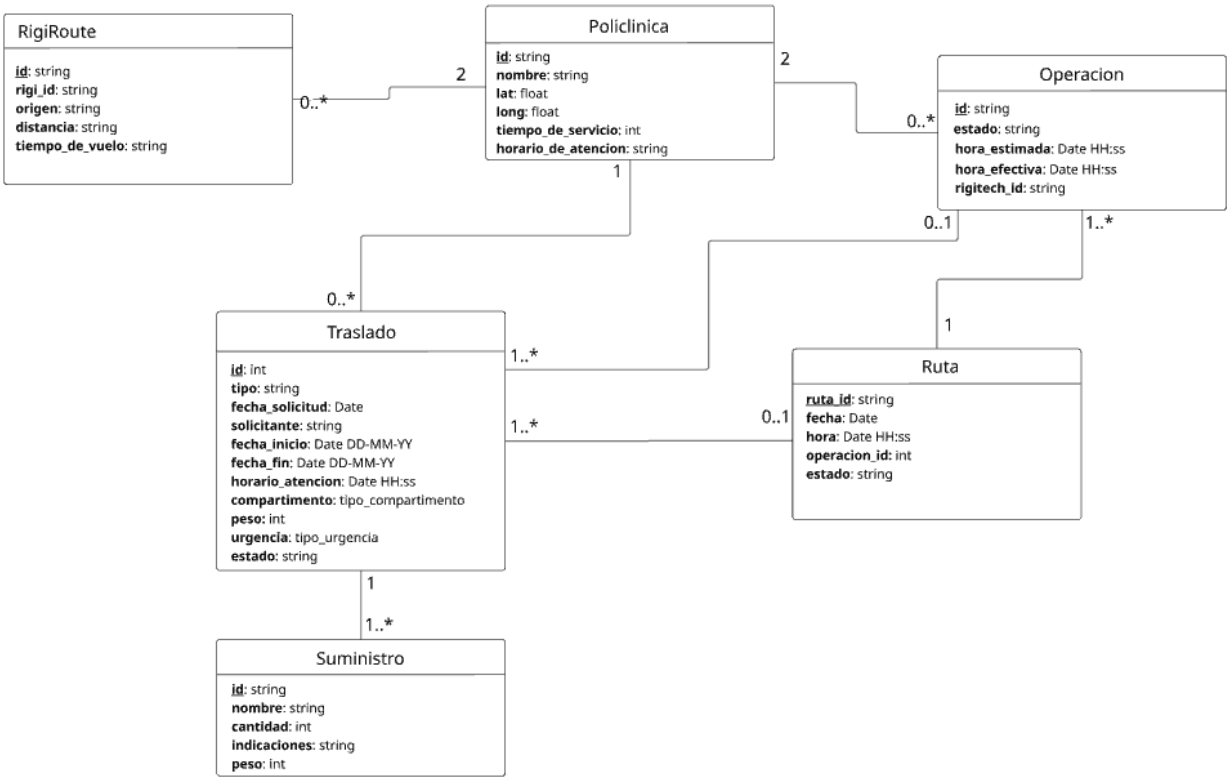

Figura 5.2: Diagrama Modelo Entidad Relación.

hora de ejecución, estado y la referencia a la operación que la ejecuta.

La entidad **Operación** refiere a una instancia de vuelo del dron. Contiene datos como identificador, estado, hora estimada, hora efectiva y el identificador externo (provisto por RigiTech).

La entidad **Rigitech Routes** almacena la información de las rutas disponibles en Rigitech, que el dron utiliza para trasladarse entre policlínicas. En esta tabla se registran datos como el origen y el destino, la distancia recorrida y la duración estimada del vuelo.

Las relaciones entre entidades fueron definidas respetando la lógica del dominio. Por ejemplo, una ruta puede incluir múltiples operaciones, una ruta puede agrupar varios traslados, y cada traslado puede involucrar múltiples suministros.

5.6. Despliegue

El sistema se compone de dos aplicaciones principales: el frontend (aplicación web) y el backend (servidor con API REST), ambos diseñados para ejecutarse en un entorno local de desarrollo o en un servidor Linux.

La aplicación web se despliega como un proceso independiente del sistema operativo. Para su ejecución, se requiere tener instalado Node.js y las dependencias del proyecto, las cuales se instalan mediante el gestor de paquetes `npm`. Una vez instaladas, se puede ejecutar el comando `npm run start`, lo que inicia un servidor de desarrollo accesible desde un navegador en `http://localhost:8080/`. Esta interfaz permite la interacción de los usuarios (enfermería o piloto) con el sistema de forma visual.

El backend, desarrollado en Python utilizando el framework Flask, se ejecuta como un servicio del sistema. Puede iniciarse manualmente mediante el comando `python run.py`. Este servicio expone una API REST en el puerto `8081`, accesible a través de solicitudes HTTP en `http://localhost:8081/`. La API maneja la lógica de negocio, la conexión con la base de datos PostgreSQL, y las solicitudes de planificación.

Ambos componentes se comunican localmente a través de HTTP. La aplicación web realiza llamadas a la API del backend para cargar o modificar traslados, obtener información de rutas, o disparar el planificador. Por su parte, el backend mantiene conexión con una instancia de PostgreSQL para almacenar y recuperar datos estructurados, como información de policlínicas, traslados, rutas y operaciones.

6. Pruebas

En esta sección se presentan las pruebas realizadas sobre el sistema desarrollado, con el objetivo de verificar su correcto funcionamiento y evaluar su desempeño en distintos escenarios. Las pruebas se dividieron en tres grandes grupos: pruebas de funcionamiento, pruebas de estrés y performance, y pruebas específicas del algoritmo de ruteo. Finalmente, se discute la posibilidad de comparar el sistema con la situación actual de los traslados.

Las pruebas fueron realizadas en un entorno local utilizando una computadora portátil Apple MacBook Pro equipada con un procesador Apple M2, 8 núcleos (4 de alto rendimiento y 4 de eficiencia), 16 GB de memoria RAM y unidad de almacenamiento SSD de 512 GB. Tanto el backend como el frontend del sistema, así como el motor de planificación, se ejecutaron en este mismo equipo. Al tratarse de un entorno de desarrollo local, no se utilizó una arquitectura cliente-servidor distribuida, sino que todas las pruebas se realizaron en una única máquina.

Si bien por el momento no se cuenta con datos suficientes para realizar una comparación en términos de optimización logística o mejoras operativas concretas, sí es posible establecer una comparación desde el punto de vista administrativo, destacando el potencial del sistema para reducir la carga de trabajo humano y los costos asociados a la planificación manual de los vuelos.

6.1. Pruebas de Funcionamiento

Las pruebas de funcionamiento se centraron en verificar que las funcionalidades principales del sistema se comportaran de acuerdo con los requisitos planteados. En primer lugar, como se muestra en la Figura 6.1, se evaluó el proceso de ingreso de un traslado desde la interfaz de usuario. Para ello, se procedió a completar un formulario con la información correspondiente a un traslado, incluyendo la policlinica solicitante, si es un envio o pedido, los suministros, la urgencia, la ventana de tiempo, el peso y el volumen del pedido.

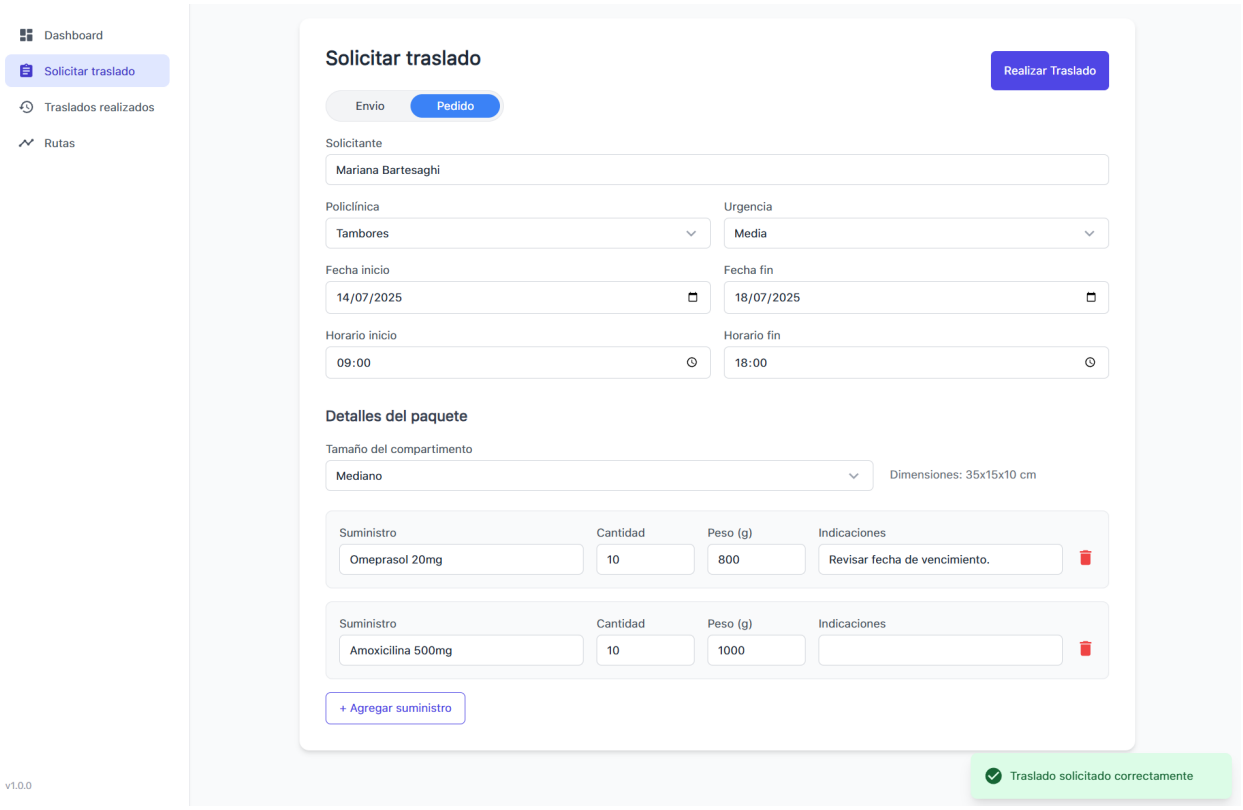

Figura 6.1: Caso de uso Solicitar traslado. Flujo exitoso.

Una vez ingresada la información, se comprobó que el sistema registrara correctamente el traslado en la base de datos, permitiendo su posterior visualización. Un dato relevante sobre estos datos es el estado inicial del traslado que se encuentra en "PENDING"(pendiente) a modo que está disponible a ser tomado por el algortimo de ruteo y ser asignado a una ruta.


```
{
  "id": "db315ee8-ea75-4165-8d2d-54063d0af5e7",
  "type": "PEDIDO",
  "request_date": "2025-07-11",
  "requester": "Mariana Bartesaghi",
  "start_date": "2025-07-14",
  "end_date": "2025-07-18",
  "start_time": "09:00:00",
  "end_time": "18:00:00",
  "compartment": "MEDIUM",
```

```
  "urgency": "MEDIUM",
  "status": "PENDING",
  "estimated_arrival_date": null,
  "estimated_arrival_time": null,
  "clinic_id": "1",
  "routine_id": null,
  "route_id": null,
  "operation_id": null
}
```


Además, se realizaron pruebas específicas para verificar el correcto funcionamiento de las validaciones implementadas en el sistema durante la creación de solicitudes de traslado. El sistema cuenta con diversas validaciones aplicadas a los distintos campos que el usuario debe completar al ingresar una solicitud.

En primer lugar, se validó que los campos obligatorios no puedan quedar vacíos, como es el caso del nombre del solicitante, las fechas y los horarios del traslado. Asimismo, se implementaron validaciones sobre la coherencia de las fechas, asegurando que la fecha de inicio no sea posterior a la fecha de finalización, lo que garantiza la consistencia temporal de la solicitud.

Por último, se verificaron las validaciones vinculadas a los suministros médicos ingresados. Tal como se mencionó anteriormente, el dron utilizado en los traslados posee una capacidad máxima de carga de 3 kg. Por lo tanto, como se muestra en la Figura 6.2, el sistema controla que la suma de los pesos de los insumos seleccionados no supere el umbral de 3.000 gramos, evitando así la creación de solicitudes inviables desde el punto de vista operativo. La figura 6.2 ilustra el funcionamiento de estas validaciones en el formulario de ingreso de solicitudes de traslado.

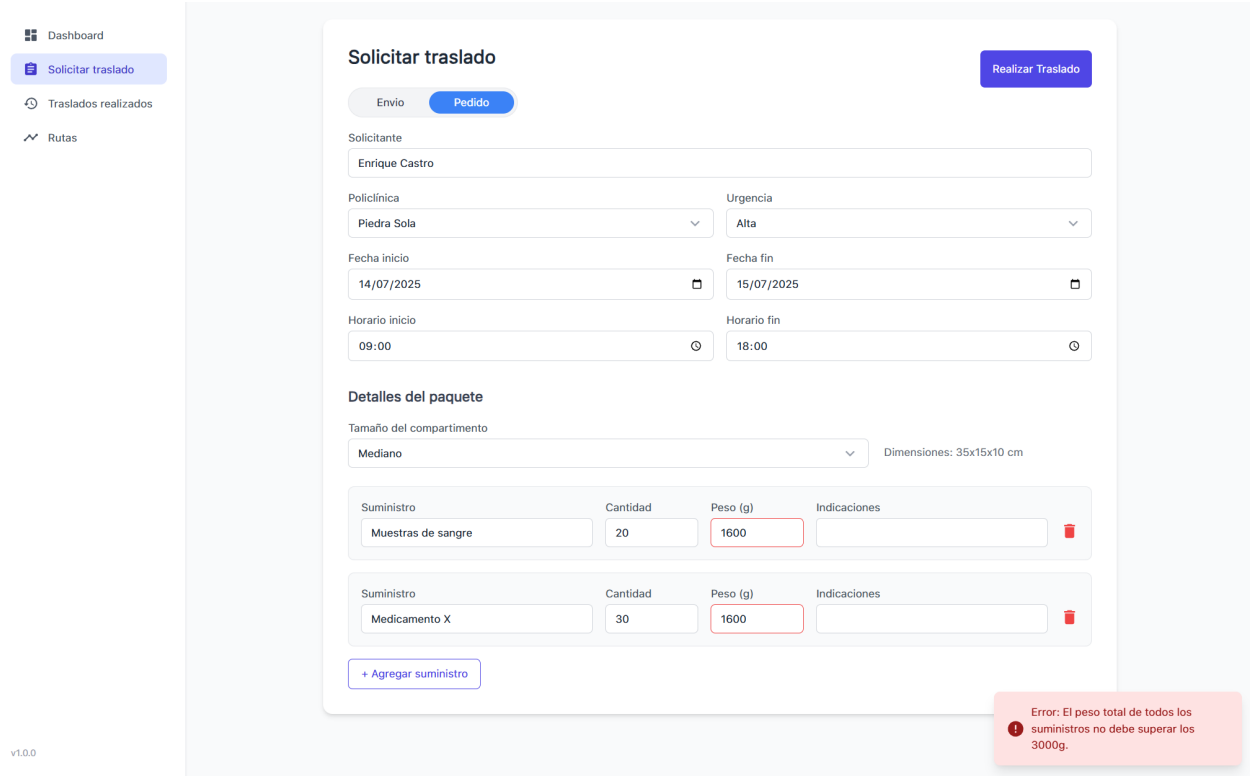

Figura 6.2: Caso de uso Solicitar traslado. Error al ingresar los datos.

La Figura 6.3 muestra una imagen del panel principal del sistema, donde se visualiza el listado de traslados pendientes. Esta pantalla permite a los usuarios acceder de forma rápida y centralizada a la información clave de cada solicitud. El objetivo de este Dashboard es facilitar la gestión y el seguimiento de los traslados que se encuentran disponibles de rutear, es decir los que están esperando ser asignados a una ruta y los que están ya confirmados en alguna.

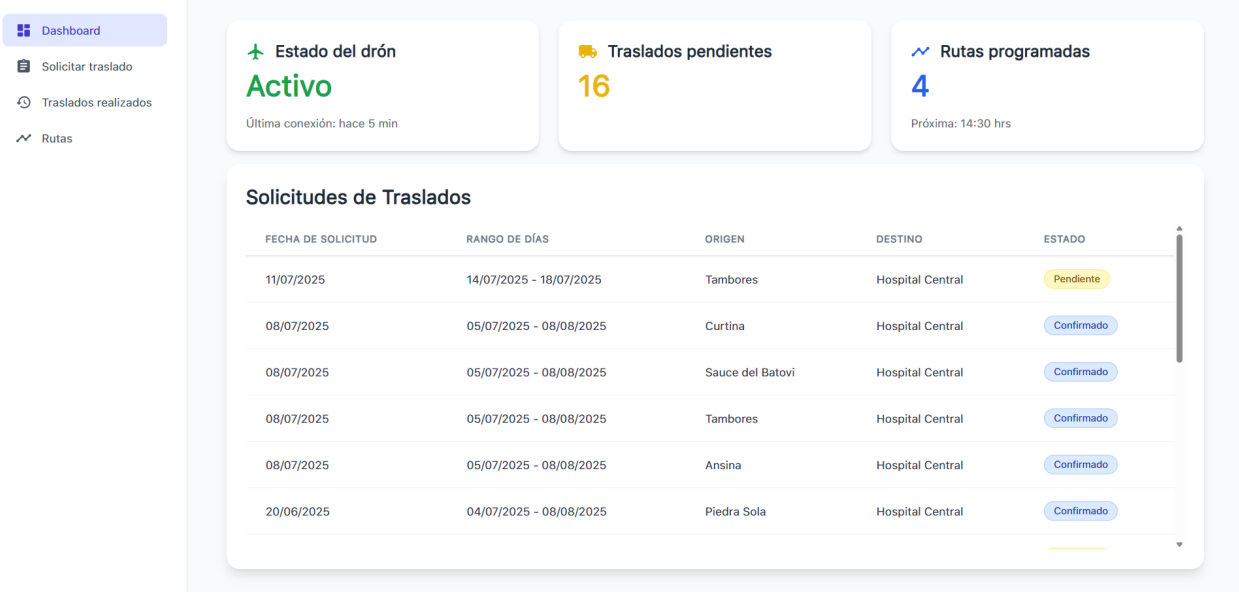

Figura 6.3: Caso de uso Visualización Dashboard Solicitudes de Traslados.

El sistema también cuenta con un panel de historial de traslados, Figura 6.4, donde es posible visualizar el registro completo de todas las solicitudes realizadas. En este historial se muestran tanto los traslados pendientes como aquellos que ya fueron confirmados, rechazados o entregados, permitiendo un control integral de cada operación.

Además, esta pantalla incorpora funcionalidades de filtrado que facilitan la búsqueda de información específica. Los usuarios pueden filtrar los traslados por texto, por estado o por rango de fechas de solicitud, lo que permite acceder de forma rápida y eficiente a los datos relevantes en función de las necesidades de consulta o seguimiento.

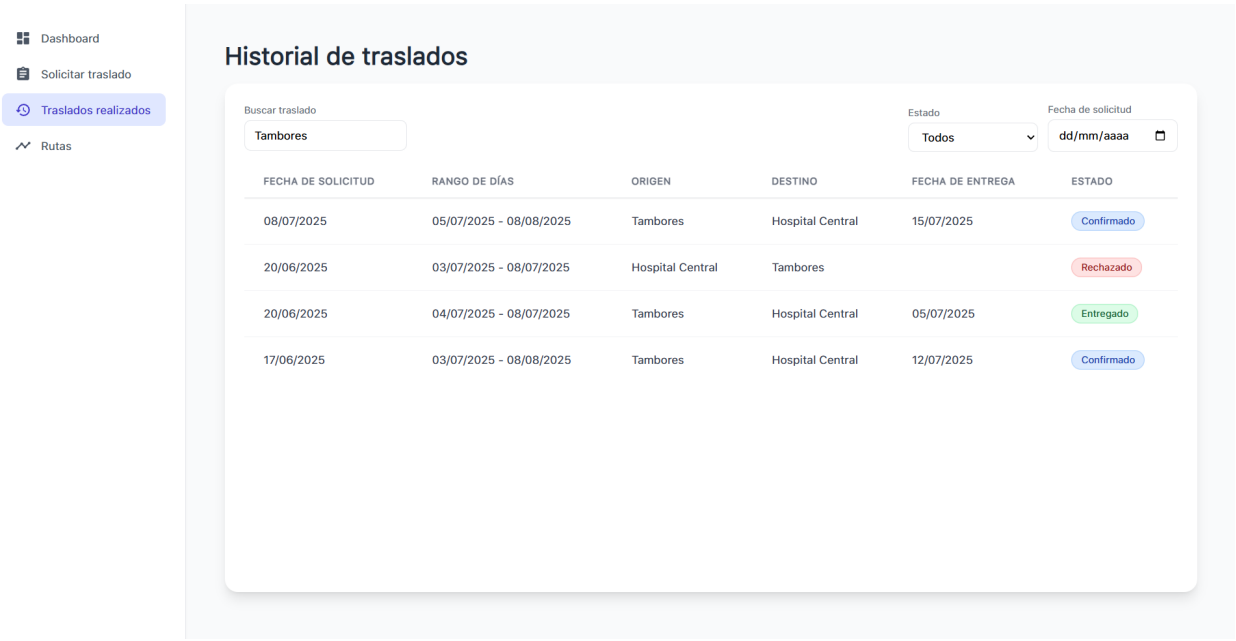

Figura 6.4: Caso de uso Visualización Dashboard de Historial de Traslados.

Posteriormente, se validó el funcionamiento de la planificación semanal de traslados, como se observa en la Figura 6.5. Se utilizó la opción del sistema que permite generar una planificación automática para un período de tiempo determinado, en este caso una semana. Al ejecutar esta funcionalidad, el sistema creó los registros de planificación correspondientes, asignando los traslados a los recursos disponibles de acuerdo con la lógica establecida por el algoritmo de ruteo. Se verificó que los traslados quedaran correctamente distribuidos a lo largo de los días planificados y que los registros pudieran ser consultados y utilizados para la ejecución de los vuelos.

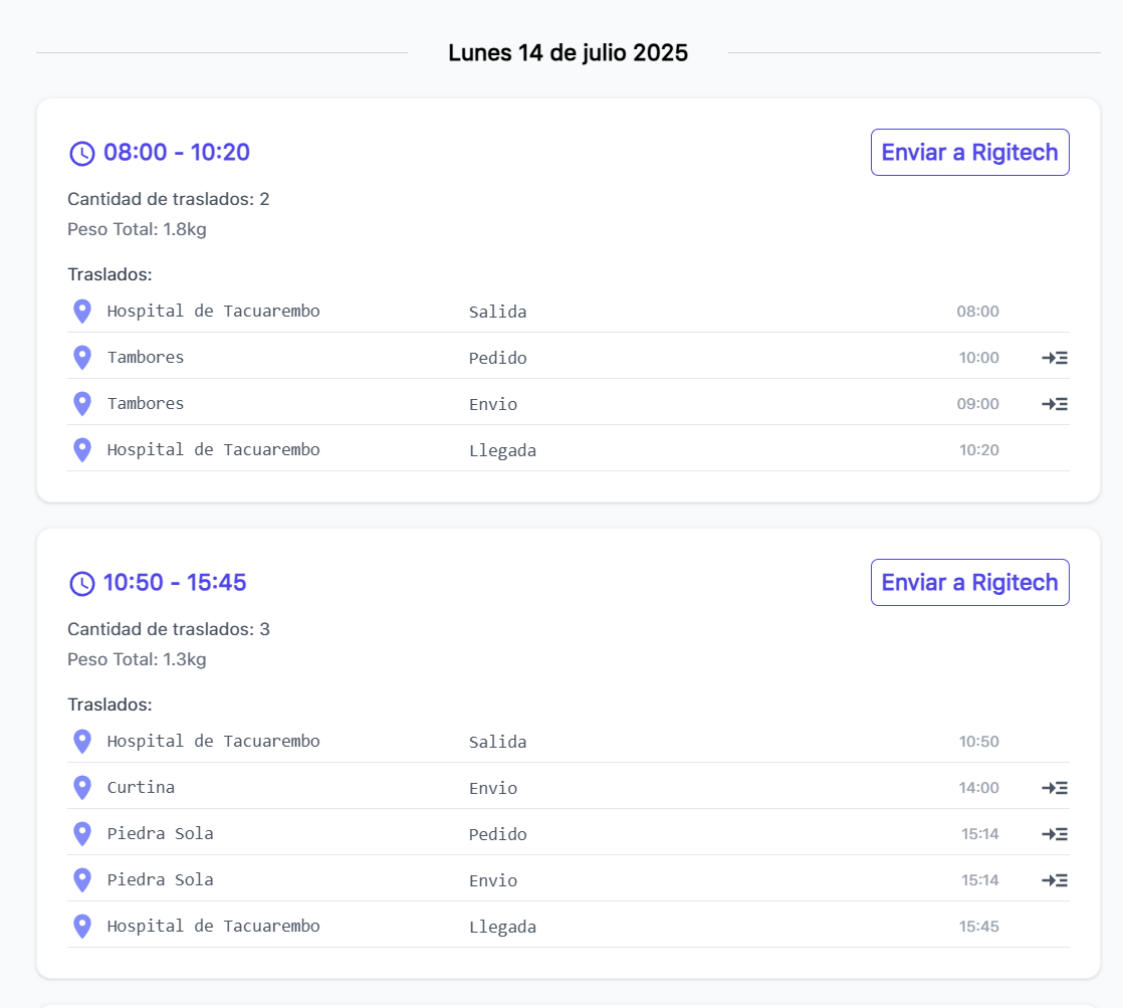

Figura 6.5: Caso de uso Visualización de Rutas Generadas.

También se incluye, por conveniencia, la posibilidad de visualizar los detalles de cada traslado mediante un botón ubicado en la parte derecha de cada registro, como se muestra en la Figura 6.6. Al hacer clic, se despliega una ventana emergente con toda la información asociada al traslado.

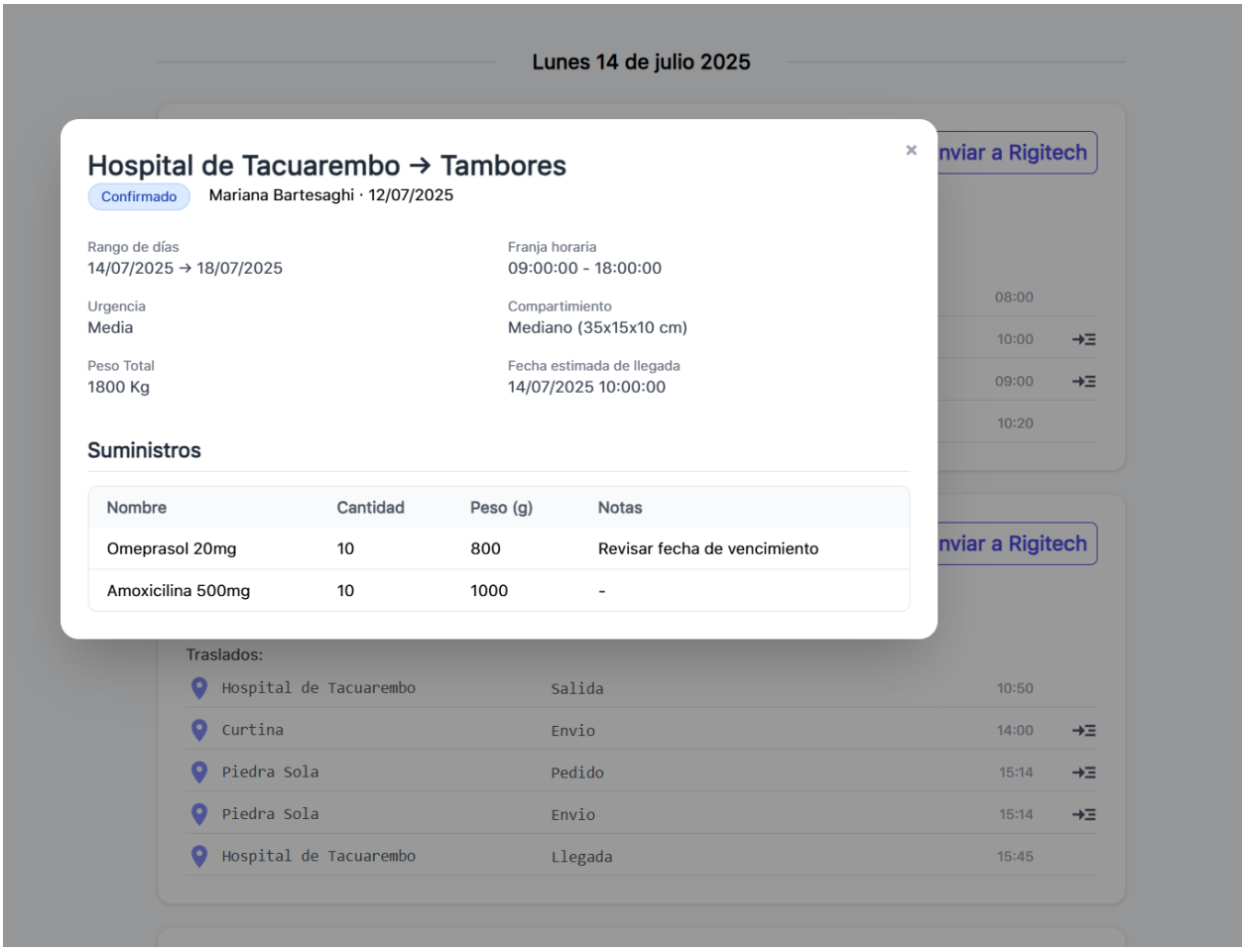

Figura 6.6: Visualización de detalles de los traslados asociados a una ruta.

Una vez que el traslado es incluido en una de las rutas generadas, su información asociada se actualiza automáticamente: el estado cambia de **PENDIENTE** a **PLANIFICADO**, y se completan campos como `estimated_arrival_date` y `estimated_arrival_time`, que indican la fecha y hora estimadas de arribo del dron a la policlínica. Además, se asigna el valor de `route_id`, que vincula el traslado con su ruta correspondiente.


```
{
  "id": "ad5393f9-1443-4905-b188-552a45370f67",
  "type": "PEDIDO",
  "request_date": "2025-07-12",
  "requester": "Mariana Bartesaghi",
```

```
  "start_date": "2025-07-14",
  "end_date": "2025-07-18",
  "start_time": "09:00:00",
  "end_time": "18:00:00",
  "compartment": "MEDIUM",
  "urgency": "MEDIUM",
  "status": "CONFIRMED",
  "estimated_arrival_date": "2025-07-14",
  "estimated_arrival_time": "10:00:00",
  "clinic_id": "1",
  "routine_id": null,
  "route_id": "ff9379ef-dc85-4b33-9901-ff667d81d816",
  "operation_id": null
}
```


Otra validación realizada consistió en comprobar que el sistema pudiera enviar correctamente la información del ruteo generado a la plataforma de RigiTech como se muestra en la Figura 6.7, de modo que esta pudiera ser incorporada al cronograma del dron y permitir así la ejecución de cada traslado como una operación individual. Para ello, cada ruta listada en el panel de Rutas Generadas cuenta con un botón que permite enviar sus datos a RigiTech y actualizar automáticamente la agenda del dron. Basta con hacer clic en dicho botón para que la información se transmita y se refleje en la plataforma de RigiTech, tal como se muestra en la siguiente imagen.

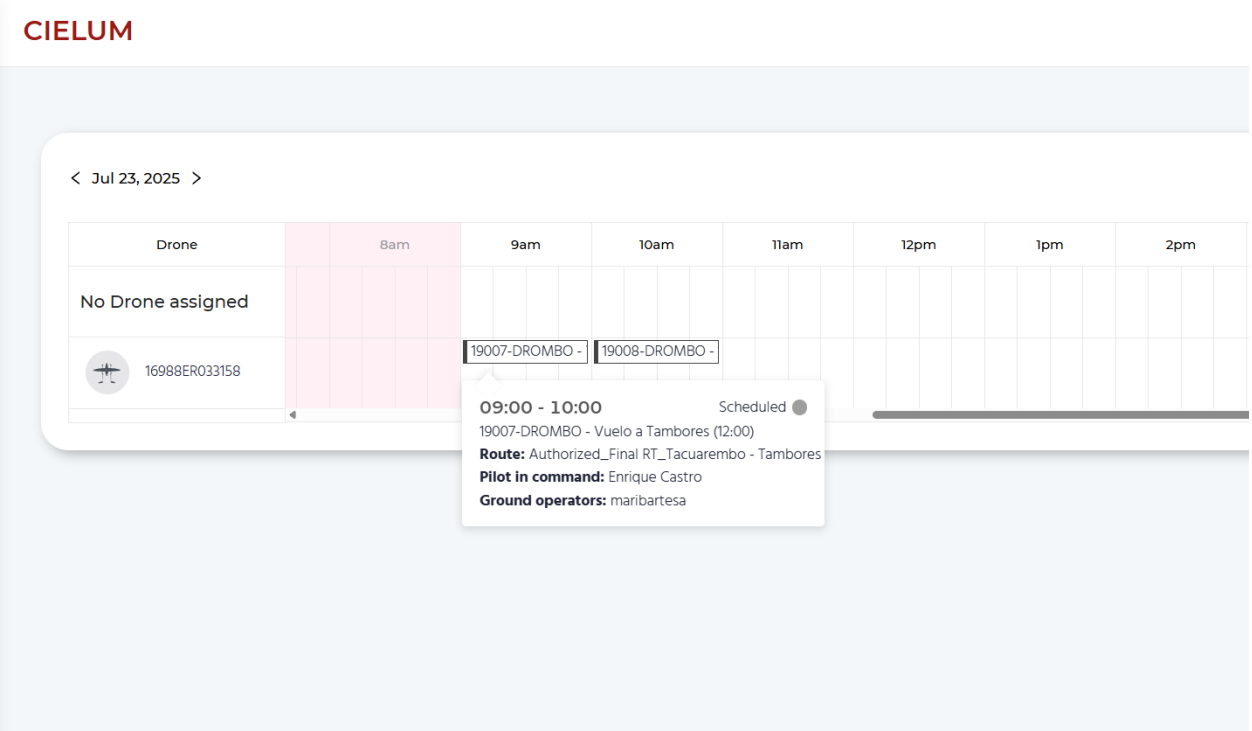

Figura 6.7: Visualización agenda de traslados del dron en la plataforma RigiTech.

Una vez ejecutada esta acción, los traslados que previamente se encontraban en estado **PLANIFICADO** dentro del sistema DROMBO pasarán a mostrarse como **CONFIRMADO**, lo que indica que la información correspondiente fue correctamente enviada y cargada en la plataforma externa.

Posteriormente, una vez que el vuelo ha sido completado, el sistema actualiza automáticamente el estado de los traslados involucrados a **ENTREGADO** y marca la ruta como **COMPLETADA**, reflejando así que la operación se ha llevado a cabo con éxito, como se muestra en la Figura 6.8. De esta forma, el sistema mantiene un historial detallado de todas las operaciones realizadas con el dron, lo que permite realizar un seguimiento preciso y contar con trazabilidad completa de cada traslado efectuado.

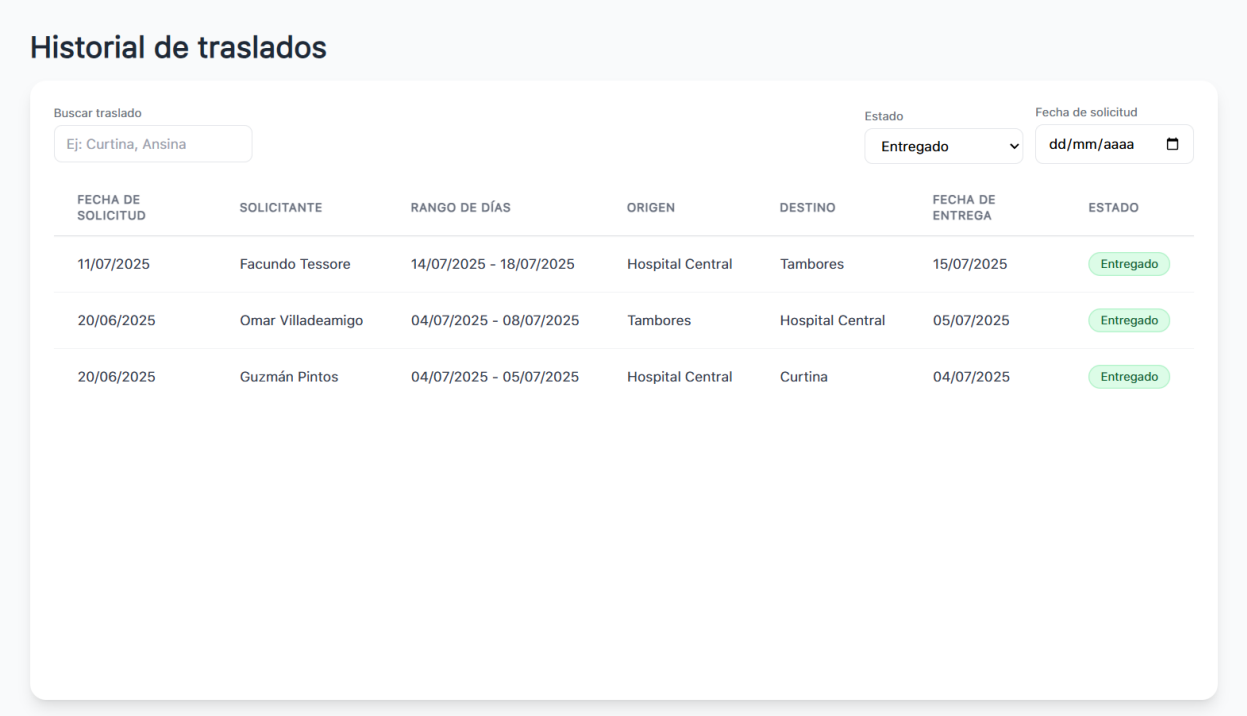

Figura 6.8: Visualización de traslados completados.

Estas pruebas permitieron confirmar que el sistema cumple con los requisitos funcionales básicos y que las principales operaciones se realizan de forma correcta y consistente.

6.2. Pruebas de Estrés y Performance

Las pruebas de estrés y performance tuvieron como finalidad evaluar la capacidad del sistema para soportar escenarios de alta demanda y su comportamiento bajo carga extrema. Para simular un uso intensivo, se planteó un escenario donde múltiples usuarios, ubicados en diferentes policlínicas, ingresan de forma simultánea un gran volumen de solicitudes de traslados. Esta simulación buscó reflejar una posible situación real en la que se produzca una alta demanda de traslados en un corto período de tiempo.

En este contexto, se realizó una prueba de carga utilizando la herramienta **k6** [97], la cual permitió emular 50 usuarios concurrentes realizando un total de 10.000 solicitudes `POST` al endpoint de creación de traslados. Cada usuario ejecutó iteraciones sobre un mismo payload

con información válida, con el objetivo de verificar la estabilidad del sistema, el tiempo de respuesta promedio y la tolerancia ante concurrencia.

Los resultados obtenidos, como se pueden visualizar en la Tabla 6.9, fueron altamente satisfactorios:

- El 100 % de las solicitudes recibieron respuesta con código `200`, sin errores ni pérdidas de información.
- El sistema logró mantener un tiempo de respuesta promedio de **273,96 ms**, con un mínimo de **29,37 ms** y un máximo de **412,7 ms**.
- El 90 % de las solicitudes fueron respondidas en menos de **299,57 ms**, y el 95 % en menos de **310,3 ms**.
- No se registraron fallos en el servidor ni caídas del servicio.
- El consumo de red fue de **10 MB recibidos** y **6 MB enviados** durante toda la prueba.

Métrica	**Valor**
Solicitudes totales	10.000
Usuarios virtuales simultáneos	50
Duración promedio por solicitud	273,96 ms
Tiempo mínimo de respuesta	29,37 ms
Tiempo máximo de respuesta	412,7 ms
Percentil 90	299,57 ms
Percentil 95	310,3 ms
Errores HTTP	0
Datos recibidos	10 MB
Datos enviados	6 MB

Tabla 6.9: Resumen de métricas obtenidas en la prueba de carga con k6

Estas métricas demuestran que el sistema es capaz de manejar cargas elevadas sin degradación significativa del rendimiento ni afectación de la estabilidad. Si bien es esperable que los tiempos de respuesta se incrementen en contextos de concurrencia intensa, los valores obtenidos se encuentran dentro de márgenes aceptables para una aplicación de este tipo. En consecuencia, el sistema se considera apto para operar en condiciones reales de alta demanda, manteniendo la integridad de los datos y el correcto funcionamiento de sus servicios principales.

6.3. Pruebas del Algoritmo de Ruteo

El algoritmo de ruteo fue evaluado mediante pruebas específicas orientadas a analizar su rendimiento frente a escenarios con una elevada carga de traslados. El objetivo principal de estas pruebas fue verificar que el sistema pudiera planificar de manera eficiente un conjunto significativo de solicitudes distribuidas a lo largo de una semana, respetando las restricciones operativas impuestas (ventanas de tiempo, urgencias, capacidad del dron, etc.).

Para esta prueba en particular, se generó un conjunto de **65 solicitudes de traslado**, cada una con fechas distribuidas dentro del rango semanal del **20 al 27 de julio de 2025**. Posteriormente, se ejecutó el algoritmo de planificación y se midió el tiempo requerido para completar el proceso de asignación de rutas. El resultado fue un tiempo de planificación de:

0.151 segundos

Este valor demuestra que el algoritmo es capaz de procesar una cantidad considerable de pedidos en un tiempo extremadamente reducido, lo cual es especialmente relevante en contextos donde se requiere una respuesta rápida y periódica durante el día.

Las rutas generadas como resultado de esta planificación se muestran en las Figuras 6.9 y 6.10. Como puede observarse, el sistema logró distribuir eficientemente los traslados a lo largo de toda la semana, agrupando múltiples pedidos dentro de una misma ruta cuando las restricciones lo permitían. Esta agrupación no solo permite aprovechar mejor la capacidad del dron, sino también reducir la cantidad total de vuelos necesarios, optimizando el uso del recurso sin necesidad de aplicar heurísticas de mejora complejas.

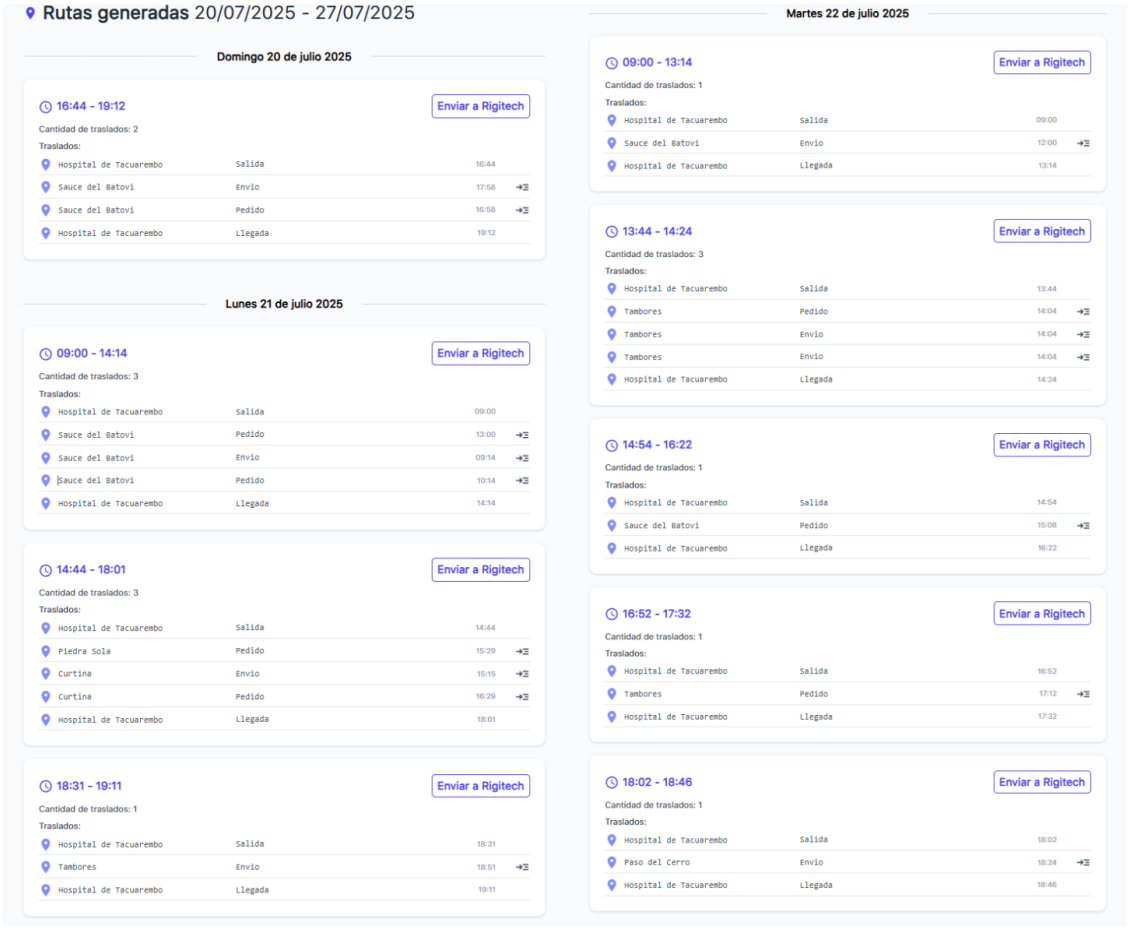

Figura 6.9: Rutas generadas por el algoritmo para los días 20 al 22 de Julio de 2025.

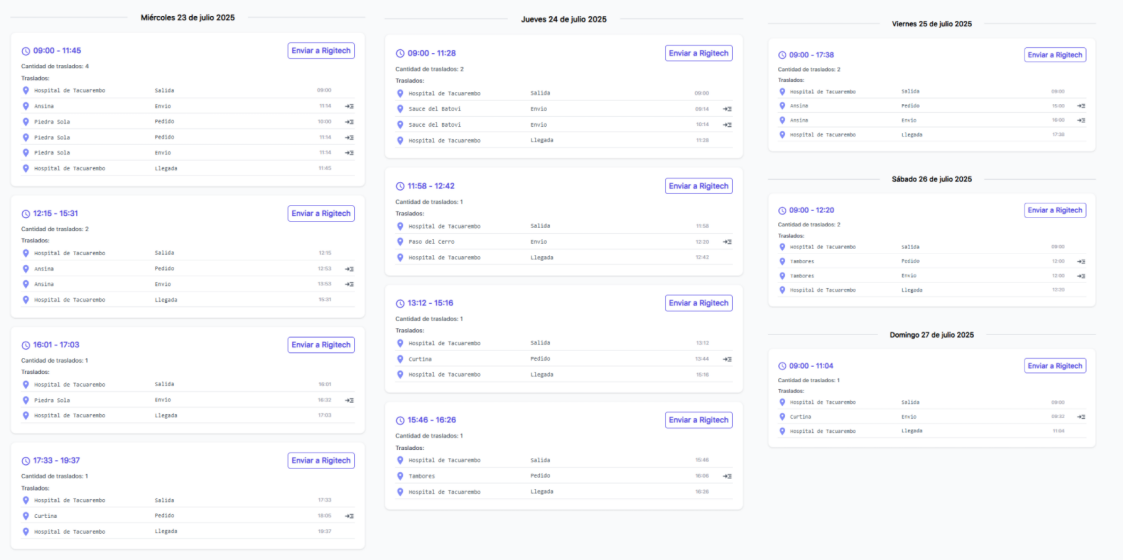

Figura 6.10: Rutas generadas por el algoritmo para los días 23 al 27 de Julio de 2025.

En total, se planificaron **18 rutas**, muchas de las cuales combinan múltiples traslados entre policlínicas diferentes, respetando siempre la capacidad máxima de carga del dron y las ventanas de tiempo establecidas. Se observa además una buena distribución de la carga operativa a lo largo de la semana, evitando concentraciones excesivas en un mismo día.

Aunque el sistema no implementa técnicas de optimización exacta, el enfoque actual logra soluciones satisfactorias desde el punto de vista logístico y operativo. Esta estrategia garantiza una planificación eficiente, simple de mantener y capaz de adaptarse dinámicamente a cambios en la demanda de traslados.

6.4. Comparación con la Situación Actual

Actualmente, la coordinación de los vuelos de traslados es gestionada manualmente por el gerente de operaciones del hospital, quien se encarga de organizar las solicitudes, definir los vuelos y coordinar la disponibilidad del piloto y los usuarios de enfermería. Este proceso requiere la dedicación permanente de al menos una persona que, además del piloto durante la operación del dron, resulta indispensable para que los traslados puedan llevarse a cabo. Este modelo de gestión no solo es dependiente de la disponibilidad humana, sino que tam-

bién implica un costo mensual estimado en aproximadamente 3.000 dólares asociados a la necesidad de mantener esta función operativa.

La implementación del sistema propuesto permitiría automatizar la planificación de los vuelos, eliminando la necesidad de una persona dedicada exclusivamente a la coordinación de los traslados. De esta forma, se lograría una mejora en la eficiencia organizativa, una reducción en los costos operativos mensuales y una mayor capacidad para escalar el uso del dron sin aumentar la carga administrativa.

En lo que respecta al análisis de la eficiencia logística derivada de la utilización del algoritmo de ruteo, particularmente en términos de ahorro de tiempo, mejora en la utilización del dron o reducción de tiempos de respuesta, no se cuenta con datos reales que permitan realizar una comparación cuantitativa con la situación actual. A lo largo del desarrollo de este proyecto no fue posible acceder a información concreta sobre cuántos vuelos se realizan por día, ya que por parte del cliente no se tiene un registro del mismo. La ausencia de estos datos impidió la posibilidad de establecer una línea de base sobre la cual comparar de forma objetiva los resultados obtenidos con el sistema propuesto.

Para que este tipo de análisis pueda llevarse a cabo en el futuro, será necesario un período de uso prolongado del sistema en un entorno real, donde se puedan registrar y medir indicadores clave como la cantidad de vuelos realizados, los tiempos de respuesta promedio, la cobertura alcanzada y la carga de trabajo del personal involucrado. Por este motivo, la cuantificación de los beneficios logísticos y operativos que ofrece el sistema se plantea como una línea de trabajo a futuro, una vez que se disponga de información empírica suficiente para realizar una comparación precisa.

7. Conclusiones y Trabajos Futuros

El presente proyecto de fin de carrera permitió desarrollar un primer paso para una solución a la planificación y gestión de traslados médicos mediante drones, adaptada a las necesidades específicas del departamento de Tacuarembó. A lo largo del proyecto se abordaron desafíos de carácter técnico, logrando construir una arquitectura robusta y funcional, acompañada de una herramienta de planificación basada en heurísticas.

Desde el punto de vista algorítmico, se estudiaron diferentes variantes del problema de ruteo de vehículos (VRP), integrando restricciones reales como la capacidad de carga del dron, las ventanas de tiempo y los niveles de urgencia médica. Se optó por implementar una heurística de inserción secuencial inspirada en Solomon (1987), debido a su buen equilibrio entre calidad de solución, bajo tiempo de cómputo y adaptabilidad al contexto. Esta decisión se basó en la necesidad de mantener un sistema eficiente y fácil de mantener, en lugar de emplear metaheurísticas más complejas que podrían resultar innecesarias para la escala actual del problema.

Desde el punto de vista arquitectónico, se diseñó un sistema con una API centralizada, respaldada por una base de datos relacional (PostgreSQL), y una interfaz web utilizable por el personal de policlínicas y pilotos de dron. Esta estructura permitió una gestión clara de entidades clave como traslados, suministros y rutas, y facilita la integración con plataformas externas como RigiTech.

Las pruebas realizadas sobre el sistema validaron su correcto funcionamiento general, mostrando que las funcionalidades centrales, como el ingreso de traslados, la validación de campos, la planificación automática, la actualización de estados y la integración con la plataforma externa se comportan de manera estable y consistente. Además, se verificó que el sistema mantiene un historial completo y trazable de cada operación, lo que mejora la transparencia y el control sobre el flujo logístico.

En cuanto a performance, las pruebas de carga demostraron que el sistema es capaz de soportar altos volúmenes de solicitudes simultáneas. La ejecución de 10.000 traslados

simulados por 50 usuarios concurrentes mostró un 100 % de éxito en las peticiones, sin errores ni pérdidas de información, y tiempos de respuesta promedio aceptables, en el orden de los 274 ms. Esto confirma la solidez de la arquitectura bajo escenarios de alta demanda.

Respecto al algoritmo de ruteo, se realizaron pruebas con 65 traslados distribuidos en una semana. El sistema logró generar 18 rutas de forma eficiente en un tiempo de planificación de solo 0,151 segundos. Las rutas resultantes mostraron una distribución lógica de los traslados, agrupando múltiples pedidos cuando era posible y respetando todas las restricciones impuestas. Esto refuerza la idoneidad de la heurística seleccionada para entornos reales, donde la velocidad de cómputo es un factor clave.

En línea con la opinión técnica del Ing. Sebastián Macías, CEO de la empresa Cielum by Dronfies, se concluye que el modelo desarrollado resulta adecuado para representar los aspectos centrales de la logística aérea en el marco del proyecto, cumpliendo así con los objetivos planteados en esta tesis. Además, se destaca el valor de la integración del software de gestión con la plataforma de RigiTech, lo que refuerza la planificación operativa de los vuelos, y la idoneidad de la interfaz de solicitud de pedidos para su implementación en policlínicas, ofreciendo a los funcionarios una herramienta práctica y funcional. Finalmente, se reconoce que el proyecto abre el camino hacia futuras líneas de investigación orientadas a un modelado más detallado con datos exactos de la red, lo que permitirá evaluar con mayor precisión la viabilidad económica de la logística médica con drones, tanto en Tacuarembó como en el contexto nacional.

Este proyecto enfrentó diversas dificultades administrativas y organizativas que prolongaron su ejecución a lo largo de casi dos años. Incluso antes del inicio formal, se produjo un cambio significativo en la composición del equipo, cuando uno de los integrantes originales fue desvinculado por falta de créditos necesarios para cursar el proyecto. Esto obligó a reorganizar el trabajo entre los dos miembros restantes y a ajustar el alcance inicialmente previsto.

Durante el desarrollo, también se presentaron diferencias de criterio entre los usuarios involucrados, ya que uno de estos pretendía cambiar el alcance del proyecto mientras que

el otro no. Esto generó indefiniciones sobre el alcance y los objetivos, y requirió invertir tiempo considerable en la alineación de expectativas. Esto fue zanjando por la intervención del supervisor llegando a un acuerdo con ambos usuarios sobre la imposibilidad de modificar el alcance y objetivos.

Por otro lado, la ejecución del software y su integración con plataformas externas se vieron afectadas por demoras en aspectos administrativos. En particular, las gestiones para acceder a la API de RigiTech dependían de la firma de un contrato de confidencialidad (NDA) por parte de los estudiantes, la facultad, la empresa cliente y la empresa externa. Este trámite, inicialmente mal gestionado tuvo como resultado, aproximadamente un año de espera para obtener acceso a los recursos necesarios para avanzar con la integración final del sistema.

Finalmente, aunque no se pudo realizar una comparación cuantitativa en términos de eficiencia logística con la situación actual debido a la falta de registros por parte del cliente, por ejemplo la cantidad de vuelos por día, tiempo de demoras desde la solicitud hasta el envío de los suministros o errores ocasionados debido a fallas en la coordinación manual entre otras métricas igualmente, se identificó un beneficio administrativo inmediato: la automatización de la planificación elimina la necesidad de una persona dedicada exclusivamente a coordinar vuelos, lo que representa un ahorro operativo importante.

Trabajos Futuros

Si bien el sistema desarrollado cumple con los objetivos planteados, existen diversas líneas de mejora que pueden explorarse en futuras etapas.

Una primera línea clave es avanzar en la evaluación empírica del impacto logístico del sistema una vez en producción. Para ello, se propone realizar un seguimiento mediante indicadores concretos como la cantidad de vuelos realizados, tiempos de respuesta promedio, volumen de traslados gestionados y eficiencia en la utilización del dron. Estos datos permitirán cuantificar los beneficios operativos alcanzados y guiar futuras mejoras.

Asimismo, se abre la posibilidad de incorporar heurísticas de mejora sobre las soluciones generadas inicialmente, como por ejemplo los métodos de 2-opt o 3-opt, que permite optimizar rutas a través de la eliminación y reinserción de segmentos, reduciendo la distancia total

recorrida o los tiempos de espera. Este tipo de técnicas puede aplicarse como etapa posterior a la inserción secuencial, buscando mejorar la calidad de la solución sin alterar el enfoque general del sistema.

Considerando un escenario donde se disponga de múltiples drones, resultaría conveniente explorar el uso de heurísticas de inserción paralelas, como la propuesta de Potvin y Rousseau, que permiten planificar rutas de manera simultánea. Esto contribuiría a una mejor coordinación entre unidades y un aprovechamiento más eficiente de los recursos disponibles.

Otra línea de desarrollo consiste en profundizar la integración con la plataforma de RigiTech, permitiendo enriquecer la visualización de los vuelos en tiempo real, incorporar alertas automáticas ante eventos críticos y habilitar ajustes dinámicos en el cronograma operativo.

También se considera relevante robustecer la trazabilidad del sistema, incorporando registros de auditoría (logs) que permitan monitorear acciones clave, errores y eventos importantes para facilitar el mantenimiento y la supervisión.

Otra línea de mejora consiste en incorporar un sistema de programación automática de vuelos rutinarios, que permita definir traslados recurrentes (por ejemplo, envíos semanales de medicamentos a determinadas policlínicas) sin necesidad de reingresar manualmente la información en cada instancia. Esta funcionalidad facilitaría la planificación a largo plazo, reduciría la carga operativa sobre el usuario y aseguraría una mayor previsibilidad en la gestión logística, especialmente en contextos donde ciertos traslados se repiten con frecuencia y en horarios fijos.

Finalmente, se propone incorporar un sistema de gestión de usuarios con control de roles y permisos, lo que permitiría administrar con mayor precisión el acceso a funcionalidades según los distintos perfiles, fortaleciendo la seguridad y gobernanza del sistema.

Referencias

[1] Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2), 254–265.

[2] Unidad Reguladora y de Control de Datos Personales (URCDP). (s.f.). Guía de drones. Recuperado de https://www.gub.uy/unidad-reguladora-control-datos-personales/comunicacion/publicaciones/guia-de-drones [Accedido: 08 de septiembre de 2025].

[3] Dhakal, S., & Karunakaran, P. (2023). Design and Implementation of Drone Technology for Medical Supplement Delivery Services in Rural Regions. *International Research Journal of Engineering and Technology*, 10(12), 545–550.

[4] Bine, L., Ruiz, L., Boukerche, A., & Loureiro, A. (2023). Drone Delivery: Why, Where, and When. In *Proceedings of the 20th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (PE-WASUN '23)*, 35–43.

[5] Rigi Technologies S.A. (2024). Eiger-03 Dron Design. Recuperado de https://rigi.tech/eiger/ [Accedido: 07 de enero de 2024].

[6] Pipeline and Hazardous Materials Safety Administration (PHMSA). (2024). Transporting Infectious Substances Safely. U.S. Department of Transportation.

[7] Rigi Tech. (2023). Eiger-03 Technical Specifications Sheet and Services. Recuperado de https://rigi.tech/wp-content/uploads/2023/09/Eiger-Technical-Spec-Sheet-Aug-2023.pdf [Accedido: Febrero de 2024].

[8] EUSPA (Agencia del Programa Espacial Europeo). (2025). ¿Qué es GNSS? Recuperado de https://www.euspa.europa.eu/eu-space-programme/galileo/what-gnss [Accedido: 15 de enero de 2025].

[9] Leonardi, M. (2018). ADS-B Anomalies and Intrusions Detection by Sensor Clocks Tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 55, 2370–2381.

[10] UAV Navigation. (2018, 3 de agosto). *UAV Navigation in depth: Magnetometer, why is it critical for UAV navigation?* Recuperado de https://www.uavnavigation.com/company/blog/uav-navigation-depth-magnetometer [Accedido: 8 de septiembre de 2025].

[11] PNI Sensor. (2023). *Understanding magnetometers and their uses.* Recuperado de https://www.pnisensor.com/understanding-magnetometers-and-their-uses/ [Accedido: 8 de septiembre de 2025].

[12] Unmanned Systems Technology. (2024). Selecting an Inertial Measurement Unit (IMU) for UAV Applications. Recuperado de https://www.unmannedsystemstechnology.com/feature/selecting-an-inertial-measurement-unit-imu-for-uav-applications/ [Accedido: Febrero de 2024].

[13] Hrisko, J. (2019). Arduino Pitot Tube Wind Speed and Airspeed Indicator – Theory and Experiments. Makers Portal. Recuperado de https://makersportal.com/blog/2019/02/06/arduino-pitot-tube-wind-speed-theory-and-experiment [Accedido: Febrero de 2024].

[14] Zona, K. (2019). NASA – Pitot–Static Tube – Speedometer. Recuperado de https://www.grc.nasa.gov/WWW/K-12/Aero2000/studweb/3-2-3i2.html [Accedido: agosto de 2024].

[15] UAV Navigation. (s.f.). UAV Navigation En Profundidad: Altíme-

tros. Recuperado de https://www.uavnavigation.com/es/empresa/blog/uav-navigation-en-profundidad-altimetros [Accedido: 5 de febrero de 2025].

[16] FLARM Technology. (2024). FLARM Website. Recuperado de https://www.flarm.com [Accedido: Febrero de 2024].

[17] Everything RF. (2023). What are C2 Links? Recuperado de https://www.everythingrf.com/community/what-are-c2-links [Accedido: Febrero de 2024].

[18] Dirección Nacional de Aviación Civil e Infraestructura Aeronáutica (DINACIA). (2014). Resolución N° 291/014: Ordenamiento básico de drones. Recuperado de https://www.dinacia.gub.uy/documento/ordenamiento-basico-de-drones [Accedido: Junio de 2024].

[19] Du, L., Li, J., Gan, H., & Leng, Y. (2022). Optimal Model and Algorithm of Medical Materials Delivery Drone Routing Problem under Major Public Health Emergencies. *Sustainability*, 14, 1–26.

[20] Shi, Y., Lin, Y., & Li, B. (2022). A Bi-objective Optimization Model for the Medical Supplies' Simultaneous Pickup and Delivery with Drones. *Computers & Industrial Engineering*, 170, 1–19.

[21] Huang, F., Zhang, X., & Su, X. (2024). Optimization of Drone Delivery Routes for E-commerce in Urban Areas. *Journal of Engineering Science and Technology Review*, 17, 165–174.

[22] Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1), 80–91.

[23] Mancini, S. (2016). A Real-Life Multi-Depot Multi-Period Vehicle Routing Problem with a Heterogeneous Fleet: Formulation and Adaptive Large Neighborhood Search Based Matheuristic. *Transportation Research Part C: Emerging Technologies*, 70, 100–112.

[24] Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science*, 43(4), 408–416.

[25] Lin, S., & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2), 498–516.

[26] Orrego, J., Ospina, D., & Toro, E. (2016). Solución al Problema de Ruteo de Vehículos con Capacidad Limitada (CVRP) usando una técnica metaheurística. *Revista EIA*, 26, 227–241.

[27] Cordeau, J.-F., Laporte, G., & Mercier, A. (2001). A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. *Journal of the Operational Research Society*, 52, 928–936.

[28] Nalepa, J., & Blocho, M. (2017). A Parallel Memetic Algorithm for the Pickup and Delivery Problem with Time Windows. In *Proceedings of ITMO University, Saint Petersburg*, 1–8.

[29] Grotschel, M. (1985). The Traveling Salesman Problem. In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, 5–31. Wiley-Interscience.

[30] Røpke, S. (2006). Heuristic and Exact Algorithms for Vehicle Routing Problems. *University of Copenhagen*, 11–20.

[31] Baldacci, R., Toth, P., & Vigo, D. (2009). Exact Algorithms for Routing Problems under Vehicle Capacity Constraints. *Annals of Operations Research*, 175, 213–245.

[32] Borcinova, Z. (2017). Two Models of the Capacitated Vehicle Routing Problem. *Croatian Operational Research Review*, 8, 463–469.

[33] Kallehauge, B., Larsen, J., & Madsen, O. (2006). Vehicle Routing Problem with Time Windows. In *Column Generation*, 69–98. Springer, Boston, MA.

[34] Desaulniers, G., Desrosiers, J., & Soumis, F. (2002). VRP with Pick-up and Delivery. In *The Vehicle Routing Problem*, 225–240. SIAM.

[35] Sun, B., Yang, Y., Shi, J., & Zheng, L. (2019). Dynamic Pick-Up and Delivery Optimization With Multiple Dynamic Events in Real-World Environment. *IEEE Access*, 7, 1–12.

[36] Montoya-Torres, J. R., López Franco, J., & Felizzola Jiménez, H. (2015). A Literature Review on the Vehicle Routing Problem with Multiple Depots. *Computers & Industrial Engineering*, 79, 115–129.

[37] Laporte, G., & Nobert, Y. (1987). Exact Algorithm for the Vehicle Routing Problem. *Annals of Discrete Mathematics*, 31, 147–184.

[38] Baker, B., & Ayechew, M. (2003). A Genetic Algorithm for the Vehicle Routing Problem. *Computers & Operations Research*, 30, 787–800.

[39] Christofides, N., & Eilon, S. (1969). An Algorithm for the Vehicle-Dispatching Problem. *Operational Research Quarterly*, 20, 309–318.

[40] Eilon, S., Christofides, N., & Watson-Gandy, C. D. T. (1974). Distribution Management: Mathematical Modelling and Practical Analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 4, 589–589.

[41] Dastghaibifard, G., Ansari, E., Sheykhalishahi, S., Bavandpouri, A., & Ashoor, E. (2008). A Parallel Branch and Bound Algorithm for Vehicle Routing Problem. In *Proceedings of the International Multiconference of Engineers and Computer Scientists*, Vol. 2, 19–21.

[42] Ralphs, T. K. (2003). Parallel Branch and Cut for Capacitated Vehicle Routing. *Parallel Computing*, 29, 607–629.

[43] Little, J. D. C., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An Algorithm for the Traveling Salesman Problem. *Operations Research*, 11, 972–989.

[44] Bernardi, S. (2021). Ramificación y poda (branch & bound). Universidad de Zaragoza, 3–73.

[45] Riojas, A. (2005). Conceptos, algoritmo y aplicación al problema de las N–reinas. Monografía académica, Universidad Nacional Mayor de San Marcos, 21–31.

[46] Clarke, G., & Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12, 568–581.

[47] Olivera, A. (2004). Heurísticas para Problemas de Ruteo de Vehículos. Instituto de Computación, Facultad de Ingeniería, UdelaR, Montevideo.

[48] Lin, S. (1965). Computer Solutions of the Traveling Salesman Problem. *The Bell System Technical Journal*, 44, 2245–2269.

[49] Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2001). A Guide to Vehicle Routing Heuristics. *Journal of the Operational Research Society*, 52, 512–522.

[50] Potvin, J.-Y., & Rousseau, J.-M. (1993). A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows. *European Journal of Operational Research*, 66, 331–340.

[51] Mole, R. H., & Jameson, S. R. (1976). A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion. *Operational Research Quarterly*, 27, 503–511.

[52] Christofides, N., Mingozzi, A., & Toth, P. (1981). Exact and Approximate Algorithms for the Vehicle Routing Problem with Backhauls. *Mathematical Programming*, 20, 95–118.

[53] Fisher, M. L., & Jaikumar, R. (1981). A Generalized Assignment Heuristic for Vehicle Routing. *Networks*, 11, 109–124.

[54] Bramel, J., & Simchi-Levi, D. (1995). A Location Based Heuristic for General Routing Problems. *Operations Research*, 43, 649–660.

[55] Beasley, J. E. (1983). Route First–Cluster Second Methods for Vehicle Routing. *Omega*, 11, 403–408.

[56] Rocha, V., & Salaberry, J. (2018). Multi Depot Vehicle Routing Problem. Instituto de Computación, Facultad de Ingeniería, UdelaR, Montevideo, 23–26.

[57] Breedam, A. V. (1995). Improvement Heuristics for the Vehicle Routing Problem Based on Simulated Annealing. *European Journal of Operational Research*, 86, 480–490.

[58] Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, 13, 533–549.

[59] Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed Optimization by Ant Colonies. In *Proceedings of the First European Conference on Artificial Life (ECAL 1991)*, 134–142.

[60] Colorni, A., Dorigo, M., & Maniezzo, V. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26, 29–41.

[61] Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). Applying the Ant System to the Vehicle Routing Problem. In *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, 285–296.

[62] Osman, I. (1993). Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem. *Annals of Operations Research*, 41, 421–451.

[63] Gendreau, M., Hertz, A., & Laporte, G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, 40, 1276–1290.

[64] Taillard, É. D. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks*, 23, 661–673.

[65] Rochat, Y., & Taillard, É. D. (1995). Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1, 147–167.

[66] Xu, J., & Kelly, J. (1996). A Network-Flow Based Tabu Search Heuristic for the Vehicle Routing Problem. *Transportation Science*, 30, 379–393.

[67] Rego, C., & Roucariol, C. (1996). A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem. In *Meta-Heuristics: Theory and Applications*, 661–675. Kluwer Academic Publishers.

[68] Rego, C. (1998). A Subpath Ejection Method for the Vehicle Routing Problem. *Management Science*, 44, 1447–1459.

[69] Toth, P., & Vigo, D. (2003). The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15(4), 333–346.

[70] Barbarosoğlu, G., & Özgür, D. (1999). A Tabu Search Algorithm for the Vehicle Routing Problem. *Computers & Operations Research*, 26, 255–270.

[71] Cordeau, J.-F., Laporte, G., & Mercier, A. (2001). A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. *Journal of the Operational Research Society*, 52, 928–936.

[72] Gendreau, M., Hertz, A., Laporte, G., & Stan, M. (1998). A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows. *Operations Research*, 46, 330–335.

[73] Holland, J. (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press.

[74] Pereira, F., Tavares, J., & Costa, P. M. E. (2002). GVR: A New Genetic Representation for the Vehicle Routing Problem. In *Proceedings of the 13th Conference on Artificial Intelligence and Cognitive Science (AICS 2002)*, 95–102.

[75] Thangiah, S. R. (1995). Vehicle Routing with Time Windows Using Genetic Algorithms. In *Application Handbook of Genetic Algorithms*, 253–277. CRC Press.

[76] Thangiah, S., Osman, I., & Sun, T. (1994). Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows.

Technical Report SRU-CpSc-TR-94-27, Computer Science Department, Slippery Rock University, 332–339.

[77] Potvin, J.-Y., & Bengio, S. (1996). The Vehicle Routing Problem with Time Windows — Part II: Genetic Search. *INFORMS Journal on Computing*, 8, 165–172.

[78] Berger, J., Barkaoui, M., & Bräysy, O. (2003). A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. *INFOR*, 41, 179–194.

[79] Tan, K. C., Lee, L. H., & Ou, K. (2001). Hybrid Genetic Algorithms in Solving Vehicle Routing Problems with Time Windows. *Asia-Pacific Journal of Operational Research*, 18, 121–130.

[80] Bräysy, O., & Dullaert, W. (2003). A Fast Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. *International Journal on Artificial Intelligence Tools*, 12, 153–172.

[81] Potvin, J.-Y., & Dubé, R. (1994). Improving a Vehicle Routing Heuristic Through Genetic Search. In *Proceedings of the International Conference on Evolutionary Computation*, 194–199.

[82] Cielum. (s.f.). Cielum. Recuperado de https://www.cielum.eu [Accedido: 5 de febrero de 2025].

[83] UNICEF Uruguay. (s.f.). UNICEF Uruguay. Recuperado de https://www.unicef.uy [Accedido: 5 de febrero de 2025].

[84] Agencia Nacional de Investigación e Innovación (ANII). (s.f.). Agencia Nacional de Investigación e Innovación. Recuperado de https://www.anii.org.uy [Accedido: 5 de febrero de 2025].

[85] Dirección Nacional de Aviación Civil e Infraestructura Aeronáutica (DINACIA). (s.f.). Dirección Nacional de Aviación Civil e Infraestructura Aeronáutica. Recuperado de https://www.dinacia.gub.uy [Accedido: 5 de febrero de 2025].

[86] Mozilla Contributors. (s.f.). JavaScript. *MDN Web Docs*. Recuperado de https://developer.mozilla.org/es/docs/Web/JavaScript [Accedido: 21 de mayo de 2025].

[87] Meta Platforms, Inc. (s.f.). React – A JavaScript Library for Building User Interfaces. Recuperado de https://react.dev/ [Accedido: 21 de mayo de 2025].

[88] Google LLC. (s.f.). Material Design. Recuperado de https://m3.material.io/ [Accedido: 21 de mayo de 2025].

[89] Tailwind Labs Inc. (s.f.). Tailwind CSS – Rapidly Build Modern Websites Without Ever Leaving Your HTML. Recuperado de https://tailwindcss.com/ [Accedido: 21 de mayo de 2025].

[90] Meta Platforms, Inc. (s.f.). Context – React. Recuperado de https://react.dev/learn/passing-data-deeply-with-context [Accedido: 21 de mayo de 2025].

[91] Axios Contributors. (s.f.). Axios – Promise Based HTTP Client for the Browser and Node.js. Recuperado de https://axios-http.com/ [Accedido: 21 de mayo de 2025].

[92] Remix Software. (s.f.). React Router: Declarative Routing for React Apps. Recuperado de https://reactrouter.com/ [Accedido: 21 de mayo de 2025].

[93] Python Software Foundation. (s.f.). Python 3 Documentation. Recuperado de https://docs.python.org/3/ [Accedido: 21 de mayo de 2025].

[94] Pallets Projects. (s.f.). Flask – The Python Micro Framework for Building Web Applications. Recuperado de https://flask.palletsprojects.com/ [Accedido: 21 de mayo de 2025].

[95] Pallets Projects. (s.f.). APScheduler. Recuperado de https://pypi.org/project/APScheduler/ [Accedido: 21 de mayo de 2025].

[96] The PostgreSQL Global Development Group. (s.f.). PostgreSQL 15 Documentation. Recuperado de https://www.postgresql.org/docs/15/index.html [Accedido: Junio de 2025].

[97] k6. (s.f.). k6 Open Source. Recuperado de https://k6.io/open-source/ [Accedido: Julio de 2025].