Inteligencia Artificial

Informe Final: Multi-Depot Vehicle Routing Problem

Eduardo Mauricio Pino Huentelaf

24 de junio de 2024

Evaluación

Código Fuente (20 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

El documento se centra en el Problema de Ruta de Vehículos con Múltiples Depósitos, que implica asignar vehículos desde diferentes depósitos a clientes para minimizar la distancia total recorrida. Se revisan diversas técnicas de optimización, desde métodos clásicos hasta enfoques modernos como algoritmos genéticos. Se presenta un detallado Modelo Matemático para abordar el problema, concluyendo con una síntesis de los hallazgos clave y sugerencias para futuras investigaciones.

1. Introducción

El *Multi-Depot Vehicle Routing Problem*, que es una extensión del clásico *Vehicle Routing Problem*, implica asignar múltiples vehículos desde diferentes depósitos a un conjunto de clientes, con el fin de minimizar la distancia total recorrida. Este problema tiene aplicaciones en la gestión de flotas de vehículos y la distribución de mercancías, entre otros campos, lo que lo convierte en un tema relevante y de interés.

Se busca abordar este desafío al proporcionar una revisión detallada de las técnicas utilizadas para resolver *MDVRP*. Se espera que al entender estas técnicas, tanto profesionales como investigadores puedan tomar decisiones más fundamentadas sobre qué métodos de optimización emplear, adaptándolos a las necesidades específicas de su contexto. Además, se pretende impulsar la innovación en este campo al identificar áreas de investigación futura y posibles direcciones para el desarrollo de nuevas estrategias y enfoques.

Comienza al proporcionar una explicación detallada de *MDVRP*, destacando su importancia en la logística empresarial. Se exploran las técnicas más relevantes utilizadas para abordar *MDVRP*, desde métodos clásicos hasta enfoques modernos como los algoritmos genéticos. Luego, se presenta el Modelo Matemático utilizado para resolver el problema, proporcionando una comprensión clara de sus componentes esenciales. Finalmente, el informe concluye resumiendo los hallazgos clave y sugiere áreas de investigación futura en el campo del *MDVRP*.

2. Definición del Problema

El problema del Enrutamiento de Vehículos con Múltiples Depósitos (*MDVRP*) surgió en 1959 con el nombre de *"The Truck Dispatching Problem"* [4], motivado por la necesidad de optimizar las rutas de camiones que entregaban gasolina desde un terminal a varias estaciones de servicio. El objetivo principal era asignar estaciones a los camiones de manera eficiente, satisfaciendo la demanda de cada estación y minimizando la distancia total recorrida.

Con el tiempo, el *MDVRP* evolucionó para incluir diversas restricciones operativas, como la capacidad limitada de carga de los vehículos, ventanas de tiempo específicas para la entrega en cada cliente, límites en la duración de las rutas y variaciones en las condiciones de tráfico y distancias. Además, la heterogeneidad de la flota en términos de capacidad, velocidad y costos añadió complejidad al problema.

Como el problema iba evolucionando, surgieron diversas dificultades relacionadas con la complejidad combinatoria y la optimización de recursos. Coordinar múltiples vehículos que parten de diferentes depósitos para atender a numerosos clientes resultó ser un desafío de asignación y secuenciación complejo. Además, factores como las capacidades de carga de los vehículos y las ventanas de tiempo [6] en las que los clientes podían ser atendidos añadieron capas adicionales de complejidad al problema.

En el MDVRP, las variables principales incluyen la asignación de vehículos a depósitos, la secuencia de visitas a los clientes y las rutas que los vehículos siguen para completar las entregas. Estas variables están sujetas a diversas restricciones, como la capacidad de carga de los vehículos y las ventanas de tiempo de los clientes.

Los objetivos del *MDVRP* son claros: optimizar la utilización de recursos, minimizar los costos operativos totales y garantizar la entrega oportuna de bienes a los clientes. Esto implica encontrar soluciones que maximicen la eficiencia de las rutas de entrega y **minimicen los tiempos de viaje** y los **costos asociados**.

A lo largo del tiempo, el *MDVRP* ha evolucionado, dando lugar a diversas variantes que abordan situaciones específicas en la logística y el transporte. Partiendo del *VRP* clásico con **un único depósito**, se desarrolló el *MDVRP* con **múltiples depósitos**, y posteriormente el *MDVRPTW*, que incorpora **ventanas de tiempo** para las entregas, reflejando así la creciente complejidad y diversidad de los desafíos logísticos.

En resumen, el *MDVRP* es un **problema complejo** y **multidimensional** que requiere enfoques innovadores para su resolución. Su estudio continuo y la búsqueda de soluciones eficientes son fundamentales para mejorar la eficiencia y la rentabilidad en una variedad de contextos industriales.

3. Estado del Arte

El Multi-Depot Vehicle Routing Problem (MDVRP) se remonta a 1959 con la formulación de *"The Truck Dispatching Problem"* [4]. En este contexto, se abordaba la optimización de las rutas de una flota de camiones encargados de la entrega de gasolina desde un terminal de almacenamiento hacia un extenso número de estaciones de servicio abastecidas por dicho terminal. El objetivo principal era asignar eficientemente estaciones a camiones, asegurando la satisfacción de las demandas de cada estación y minimizando la distancia total recorrida por la flota.

El método propuesto en este inicio se basaba en una formulación de programación lineal y se dividía en múltiples etapas de agregación, en las cuales se llevaban a cabo sub-optimizaciones en pares de puntos o grupos. La determinación del número de etapas de agregación se realizaba a partir de la relación entre la capacidad de los camiones y las demandas de las estaciones.

El procedimiento computacional implicaba una resolución iterativa del problema, comenzando con una asignación inicial de estaciones a camiones y posteriormente realizando correcciones

rápidas para mejorar la solución. Para este fin, se empleaban funciones delta para decidir qué entradas no básicas aceptar en cada iteración. Además, se discutía cómo manejar soluciones fraccionarias que pudieran surgir durante el proceso de optimización.

Este problema inicial evolucionó con el tiempo, dando lugar a variantes que consideraban problemas con múltiples productos en cada estación y la capacidad variable de los camiones. Esta diversificación reflejaba la creciente complejidad y variedad de situaciones encontradas en el ámbito del transporte y la logística.

Ya con el paso de los años, comenzaron a implementarse *"Vehicle Routing Algorithms"* [1], específicamente en el contexto de la distribución de periódicos en áreas urbanas en el año 1977. La necesidad de mejorar la eficiencia en la entrega de periódicos a cientos de puntos de demanda en una ciudad se debe a varios factores que impactan la logística de distribución. Algunas de las razones de importancia para optimizar este proceso son: reducción de costos en la distribución, mejora de la eficiencia operativa en el menor tiempo posible y reducción de emisiones e impacto ambiental.

La eficiencia en el procesamiento de datos es esencial para garantizar tiempos de ejecución razonables y obtener resultados óptimos en la optimización de rutas de distribución. Por lo tanto, usar **algoritmos heurísticos** para abordar estos problemas con conjuntos masivos de datos será primordial, permitiendo encontrar soluciones aproximadas en un tiempo razonable.

Esto irá acompañado de utilizar **estructuras de datos adecuadas**, como los **árboles o grafos**, lo que puede mejorar significativamente la eficiencia en la manipulación de los datos. Organizar la información de manera que las operaciones de búsqueda, inserción y eliminación se realicen de forma eficiente es fundamental.

Por otro lado, la capacidad de manipular y operar conjuntos masivos de datos de manera efectiva, **tomando decisiones informadas basadas en análisis detallados de la información disponible**, es crucial para obtener resultados precisos en la optimización de rutas de distribución y en la toma de decisiones logísticas. Esta combinación de algoritmos heurísticos eficientes y estructuras de datos adecuadas sienta las bases para abordar con éxito los desafíos logísticos en el ámbito del *Vehicle Routing* a gran escala.

Luego, en la década de 1980, los investigadores Gilbert Laporte, Yves Nobert y Serge Taillefer se embarcaron en un desafío apasionante [5]. En medio de los avances tecnológicos y la creciente demanda de soluciones eficientes, estos investigadores se sumergieron en la complejidad de problemas de enrutamiento asimétricos en entornos *Multi-Depot*.

Utilizando un método de árbol de ramificación y poda, transformaron estos desafíos en problemas de asignación restringida, buscando incansablemente soluciones óptimas que mejoraran la planificación logística y redujeran costos operativos. En un contexto donde la logística es la columna vertebral de la cadena de suministro, este estudio representa un hito en la investigación de optimización de rutas y ubicaciones.

La capacidad para resolver problemas que involucran hasta 80 nodos es destacada. Laporte, Nobert y Taillefer ofrecieron un método de árbol de ramificación y poda, permitiendo resolver problemas de tamaño considerable, lo que demuestra la eficacia y escalabilidad del enfoque propuesto. Su alcance se destaca en su aplicabilidad en entornos logísticos complejos, donde la optimización de rutas, vehículos y ubicaciones de depósitos es crucial.

Al poder abordar problemas de enrutamiento asimétricos en entornos multi-depósito, esta metodología ofrece una solución valiosa para empresas y organizaciones.

Una década después, en 1992, Jorge Orestes Cerdeira realizó una formulación matemática detallada del *Multi-Depot Vehicle Routing Problem* [3], donde se define un grafo ponderado no dirigido $G = (V, E, c)$, con V representando el conjunto de nodos (clientes y depósitos), E como el conjunto de aristas con costos asociados, y c como la función de costo de viajar entre nodos. Se establecen restricciones sobre cómo asignar vehículos a depósitos, cómo incluir nodos en ciclos

específicos, y cómo minimizar la suma de los costos de las aristas en los ciclos determinados, lo que hace que este problema presente una complejidad *NP-Hard* debido a su naturaleza combinatoria y la presencia de múltiples depósitos y clientes, requiriendo así el desarrollo de algoritmos eficientes y heurísticas para su resolución efectiva.

Estos algoritmos se basan en técnicas que involucran la cobertura de nodos mediante ***trees and matchings***, lo que permite establecer límites inferiores en el costo de soluciones óptimas. Además, presentan algoritmos de tiempo polinómico que garantizan que los costos de **las soluciones obtenidas no excedan el doble de los costos óptimos**, lo cual es fundamental para la optimización de las rutas de vehículos desde múltiples depósitos hacia los clientes. Se llevan a cabo pruebas utilizando grafos de distintos tamaños y se consideran matrices de costos que cumplen con la desigualdad triangular, simplificando así la resolución del problema. Se comparan los límites inferiores obtenidos mediante un método de subgradiente [10] con los costos de las soluciones generadas por el algoritmo propuesto por *J.O. Cerdeira*, lo que permite evaluar la eficiencia de los enfoques utilizados en la optimización del *MDVRP*.

Un par de años más tarde, Jacques Renaud, Gilbert Laporte y Fayez Boctor propusieron un enfoque innovador para resolver el MDVRP, utilizando una técnica de optimización conocida como **Tabu Search** [8]. Esta estrategia se basa en explorar el espacio de búsqueda moviéndose de una solución a su mejor vecino, incluso si esto implica una disminución en la función objetivo.

El algoritmo propuesto por este trío de investigadores se divide en dos partes principales: la construcción de una solución inicial y la aplicación de *tabu search*.

En la fase de construcción de la solución inicial, cada vértice se asigna a su depósito más cercano y se aplica una heurística de VRP mejorada para el conjunto de vértices de cada depósito. Se utiliza el algoritmo de Petal Mejorado de Renaud [7] para generar conjuntos de rutas que pueden ser atendidas por uno o dos vehículos, seleccionando la mejor ruta mediante la resolución de un problema de partición de conjuntos.

La aplicación de *tabu search* consta de tres fases: Mejora Rápida, Intensificación y Diversificación. Cada fase utiliza procedimientos básicos de 1-ruta, 2-ruta y 3-ruta para mejorar las soluciones. Por ejemplo, el procedimiento 1-ruta se utiliza como post-optimizador en rutas de un solo vehículo, aplicando un mecanismo de mejora 4-opt* [9] para un circuito Hamiltoniano.

Los resultados de aplicar *tabu search* en 23 instancias mostraron que supera a otras heurísticas existentes en términos de calidad de las soluciones encontradas. Esto respalda la eficacia y robustez del enfoque *tabu search* para resolver *MDVRP*, destacando su capacidad para mejorar continuamente las soluciones a lo largo de las diferentes fases del algoritmo.

Con el paso de los años, comenzaron a aparecer variantes del *MDVRP*. Al igual que el *MDVRP* surge de *VRP*, en el año 2002 surgió *Multi-Depot Vehicle Routing Problem with Time Windows* [6]. Donde ahora al problema se le agrega la restricción que cada cliente tiene su horario específico en el que pueden recibir la entrega, lo que agrega una capa adicional de complejidad a la planificación de rutas. Además, los gerentes logísticos se enfrentan al desafío de asignar clientes a los depósitos más cercanos, considerando las capacidades del vehículo y respetando las ventanas de tiempo de los clientes.

Para abordar esta nueva variante del problema original, se recurren a seis heurísticas para la asignación de clientes a depósitos:

Comenzando con la heurística ***Cluster First Route Second***, esta estrategia agrupa los clientes por depósito antes de diseñar las rutas de los vehículos. Su principal ventaja radica en mejorar la eficiencia en la asignación de clientes y en la planificación de rutas. Sin embargo, una posible desventaja es que puede generar agrupamientos subóptimos dependiendo de la estrategia de agrupación utilizada.

En cuanto a la heurística ***Random Assignment***, esta asigna aleatoriamente a los clientes a los depósitos sin considerar criterios específicos. Si bien es simple de implementar y entender, puede producir asignaciones aleatorias ineficientes.

Por otro lado, la heurística ***Greedy Assignment*** se basa en un enfoque codicioso para asignar clientes a depósitos, con el objetivo de maximizar algún criterio de optimización, como la minimización de la distancia total recorrida por los vehículos. Aunque es rápida y fácil de implementar, puede quedar atrapada en óptimos locales, lo que podría ser una desventaja.

La estrategia ***Saving Assignment*** utiliza el concepto de ahorro de distancia para buscar asignaciones que generen reducciones significativas en la distancia total recorrida por los vehículos, lo que optimiza la eficiencia de las rutas. Sin embargo, requiere cálculos adicionales para determinar los ahorros de distancia, lo que podría ser una desventaja.

Por su parte, la heurística ***Nearest Depot Assignment*** asigna a cada cliente al depósito más cercano en términos de distancia, minimizando los tiempos de viaje y simplificando la planificación de rutas. Aunque es efectiva en la minimización de los tiempos de viaje, puede no tener en cuenta otras consideraciones importantes, como la capacidad del depósito.

Finalmente, la heurística ***Capacity-Constrained Assignment*** tiene en cuenta las capacidades de los depósitos al asignar clientes, garantizando una distribución equilibrada de la carga de trabajo. Sin embargo, puede ser más compleja de implementar debido a la consideración de las capacidades del depósito.

Donde de todas las heurísticas, se observó que las heurísticas que **asignaban clientes a depósitos a través de urgencias** fueron las que obtuvieron los mejores resultados computacionales. Estas estrategias basadas en urgencias demostraron un desempeño casi tan bueno como las heurísticas que **asignaban clientes por *clusters***. Por otro lado, la heurística de **asignación cíclica** fue identificada como la que obtuvo los peores resultados en términos computacionales.

En los años posteriores, comenzó a ganar popularidad la aplicación de algoritmos genéticos (*GA*) para abordar el *MDVRP*, aproximadamente desde el año 2009. En este contexto, Beatrice Ombuki-Berman y Franklin Hanshar plantearon esta aplicación [2], con la generación inicial de una población de posibles soluciones de forma aleatoria. Cada una de estas soluciones, representadas como cromosomas, se convierte en un conjunto de rutas mediante un planificador específico. Luego, se evalúa la calidad de cada solución en función de su aptitud, calculando el promedio de aptitud de toda la población.

El proceso evolutivo se inicia una vez que se tiene la generación inicial de población. En esta etapa, los cromosomas son sometidos a operaciones de cruce y selección, siguiendo los principios de los *GAs* convencionales. Además, se introduce una mutación adaptativa entre depósitos para mejorar las asignaciones iniciales de clientes a depósitos. Para la selección de individuos y la posterior reproducción, se utiliza un modelo de selección de torneo con retención de élite, priorizando a los individuos más aptos.

Un aspecto crucial es la aplicación de un operador de cruce específico del problema que garantiza que las soluciones generadas a través de la evolución genética sean todas factibles para el *MDVRP*. El proceso de optimización incluye la selección de padres, la recombinación mediante cruce, la posibilidad de mutación intra-depósito o inter-depósito, y la aceptación de nuevos descendientes en la población en lugar de los padres originales.

El proceso evolutivo continúa iterativamente hasta que se alcanza un número predeterminado de generaciones o se cumple una condición de terminación. Al final, se devuelve la aptitud promedio y la aptitud de la mejor solución encontrada en la población final como resultado del algoritmo genético. Este enfoque permite encontrar soluciones óptimas o cercanas a óptimas, mejorando la eficiencia y calidad de las soluciones obtenidas.

La comparación de los resultados del algoritmo genético propuesto para *MDVRP* mostró que el *GA* fue competitivo y en algunos casos superó a otros enfoques existentes, como el *GenClust GA* y *Tabu Search*. En general, el **algoritmo genético propuesto se destacó por su eficiencia y calidad de las soluciones**, posicionándolo como una herramienta valiosa para abordar *MDVRP* y resaltando la importancia de la investigación continua en algoritmos genéticos para problemas logísticos complejos.

Para trabajo futuro, se sugiere la necesidad de seguir investigando y desarrollando *GA* es-

pecíficamente para *MDVRP*. Se plantea la exploración de nuevas estrategias y enfoques en el diseño de algoritmos genéticos, con el objetivo de mejorar la calidad de las soluciones y competir con enfoques basados en *Tabu Search* u otras metaheurísticas.

Además, se menciona la posibilidad de considerar múltiples objetivos de optimización en el diseño de *GA* para abordar de manera más integral los desafíos asociados con la optimización de rutas de vehículos en entornos logísticos complejos.

4. Modelo Matemático

Para el desarrollo del modelo matemático, se llevó a cabo una investigación sobre qué modelo matemático podría ser más adecuado para resolver el *MDVRP*. Tânia Rodrigues Pereira Ramos, Maria Isabel Gomes y Ana Paula Barbosa Póvoa realizaron una investigación comparativa de distintos modelos matemáticos y los rendimientos computacionales que obtenían [11]. De las 4 formulaciones matemáticas, la mejor resultó ser ***Two-commodity flow formulation***, por lo que se optó por utilizar su modelo matemático, el cual será descrito a continuación:

4.1. Constantes

4.1.1. Índices

i, j: Índice de nodo
k: Índice de vehículo

4.1.2. Conjuntos

$\bar{V} = \{1, ..., n + 2w\}$: Conjunto de nodos ; $\overline{V} = V_c \cup V_d \cup V_f$

$V_c = \{1, ..., n\}$: Subconjunto de nodos de clientes

$V_d = \{n + 1, ..., n + w\}$: Subconjunto de nodos de depósitos reales

$V_f = \{n + w + 1, ..., n + 2w\}$: Subconjunto de nodos de depósitos copias

$K = K_1 \cup ... \cup K_i$

K_i: Subconjunto de vehículos pertenecientes al depósito i

4.1.3. Parámetros

d_{ij}: Distancia entre los nodos i y j

r_{ij}: Tiempo de viaje desde el nodo i al nodo j

Q_k: Capacidad del vehículo k

p_i: Demanda del cliente i

t_i: Duración del servicio al cliente i

T_i: Tiempo máximo permitido de una ruta que comienza y termina en el deposito i

s_i: Orden de visita del cliente i en la ruta

4.2. Variables

$$x_{ijk} = \begin{cases} 1, & \text{si el sitio } j \text{ se visita inmediatamente después del sitio } i, \text{ por el vehículo } k \\ 0, & \text{en caso contrario} \end{cases}$$

y_{ijk}: Variable de flujo, representa la carga del vehículo cuando el vehículo k viaja de i a j.

El flujo y_{jik} representa el k espacio vacío del vehículo; por lo tanto $y_{ijk} + y_{jik} = Q_k$

$$z_{ik} = \begin{cases} 1, & \text{si el sitio } i \text{ es visitado por el vehículo } k \\ 0, & \text{en caso contrario} \end{cases}$$

4.3. Función objetivo

$$Min(F) = \frac{1}{2} \sum_{i \in \bar{V}} \sum_{j \in \bar{V}} \sum_{k \in K} x_{ijk} d_{ij} \tag{1}$$

Minimizar la distancia total recorrida. Desde dos caminos al definir rutas, cada borde de la solución se cuenta dos veces, duplicando la distancia recorrida. Por lo tanto, para identificar la distancia real, la función objetivo tiene que dividir por 2 para eliminar la distancia del segundo camino.

4.4. Restricciones

La función objetivo está sujeta a las siguientes restricciones:

$$\sum_{j \in \bar{V}} (y_{jik} - y_{ijk}) = 2 p_i z_{ik} \quad \forall i \in V_c, \forall k \in K \tag{2}$$

(2) Establece que la entrada menos la salida de cada cliente es igual a el doble de la demanda de cada cliente.

$$\sum_{i \in V_d} \sum_{j \in V_c} \sum_{k} y_{ijk} = \sum_{j \in V_c} p_j \tag{3}$$

(3) Asegura que la salida total de depósitos reales sea igual a la demanda total del cliente.

$$\sum_{i \in V_d} \sum_{j \in V_c} \sum_{k} y_{jik} \leq \sum_{k} Q_k - \sum_{j \in V_c} p_j \tag{4}$$

(4) Dado que la capacidad de los vehículos puede exceder necesidades de los clientes, dejando algunos vehículos sin usar, la restricción (4) asegura que la entrada total de bienes reales a depósitos es, como máximo, la capacidad residual del parque de vehículos. La salida total de cada depósito de copias. Corresponde a la capacidad del parque de vehículos, con base en el depósito real correspondiente.

$$\sum_{j \in V_c} y_{ijk} \leq Q_k \quad \forall i \in V_f, \forall k \in K_i \tag{5}$$

(5) Establece que la carga total que cada vehículo al salir perteneciente al depósito de copias i es menor o igual a la capacidad máxima de carga ese vehículo. Si no se utiliza un vehículo, su

carga se establece en cero; si se utiliza un vehículo, su carga puede tomar cualquier valor hasta su capacidad máxima de ese vehículo.

$$\sum_{i \in \bar{V}} x_{ijk} = 2z_{jk} \quad \forall j \in V_c, \forall k \in K \tag{6}$$

(6) Garantiza que cualquier solución factible contenga dos aristas incidentes a cada cliente.

$$y_{ijk} + y_{jik} = Q_k x_{ijk} \quad \forall i \in \bar{V}, \forall j \in \bar{V}, \forall k \in K \tag{7}$$

(7) Garantiza que el flujo de entrada más el flujo de salida de cualquier nodo sea igual la capacidad del vehículo que visita el nodo.

$$\sum_{k \in K} z_{ik} = 1 \quad \forall i \in V_c \tag{8}$$

(8) Cada cliente deberá ser visitado por un único vehículo.

$$y_{ijk} \leq BigM z_{ik} \quad \forall i \in V_c, \forall j \in \bar{V}, \forall k \in K \tag{9}$$

(9) Pone a cero la variable de flujo y_{ijk} si el cliente i no es visitado por el vehículo k.

$$\sum_{i \in V_c} t_i x_{ijk} + \sum_{i \in \bar{V}} \sum_{j \in \bar{V}} r_{ij} x_{ijk} \leq 2T_i \quad \forall k \in K \tag{10}$$

(10) Garantiza que la duración de cada ruta (incluido el servicio y el tiempo de viaje) no exceda el tiempo máximo de enrutamiento permitido para ese depósito.

$$\sum_{j \in V_c} x_{ijk} \leq 1 \quad \forall i \in V_d, \forall k \in K_i \tag{11}$$

(11) Asegura que cada vehículo salga de su depósito de origen una vez como máximo.

$$\sum_{i \in V_c} x_{ijk} = 0 \quad \forall j \in V_f, \forall k \notin K_j \tag{12}$$

$$\sum_{j \in V_c} x_{ijk} = 0 \quad \forall i \in V_d, \forall k \notin K_i \tag{13}$$

(12) y (13) Garantizan conjuntamente que un vehículo no pueda salir y regresar a un depósito que no sea su domicilio (tanto real como copia).

$$y_{ijk} \geq 0, x_{ijk} \in \{0,1\}, z_{ik} \in \{0,1\} \quad \forall i, j \in \overline{V}, k \in K \tag{14}$$

(14) Los dominios de las variables se dan en la restricción.

$$s_i = \begin{cases} 1, & \text{donde } d_{0i} = \text{mín}_{j \in V_c}\, d_{0j} \\ s_j + 1, & \text{donde } d_{ji} = \text{mín}_{k \in V_c, s_k = 0}\, d_{jk} \end{cases} \tag{15}$$

(15) Establecer el orden de visita de los clientes: primero seleccionamos el cliente inicial como aquel que minimiza la distancia al depósito de origen. Si no es el cliente inicial, buscamos el siguiente cliente a visitar como aquel que minimiza la distancia al cliente visitado anteriormente.

5. Representación

La representación de *Multi-Depot Vehicle Routing Problem* se realiza en un plano cartesiano bidimensional con ejes x e y, donde se posicionan los clientes y depósitos como puntos. Cada cliente y depósito se asignan coordenadas específicas (x, y), lo cual facilita la visualización intuitiva de cómo los vehículos se desplazan entre los depósitos y los clientes para cumplir con las restricciones del problema. Además de los puntos, las rutas entre depósitos y clientes se representan mediante conexiones directas que muestran las trayectorias recorridas.

Esta representación no solo permite calcular de manera efectiva la distancia total recorrida por los vehículos, sino que también simplifica la generación y evaluación de soluciones óptimas. La posición de cada punto en el plano refleja su ubicación física o geográfica relativa, proporcionando una perspectiva clara de cómo se organiza y se ejecuta la distribución de vehículos.

La elección de esta representación es apropiada para *MDVRP* por varias razones. Primero, reduce significativamente el espacio de búsqueda al asignar coordenadas a cada entidad del problema. Segundo, facilita la visualización directa de las soluciones potenciales y optimiza la planificación de rutas eficientes. Además, esta representación es clara y accesible, lo que permite una comprensión inmediata de las estrategias empleadas para resolver el problema.

A continuación, se muestra un esquema que ilustra cómo se representarían depósitos, clientes y rutas entre ellos en un plano cartesiano:

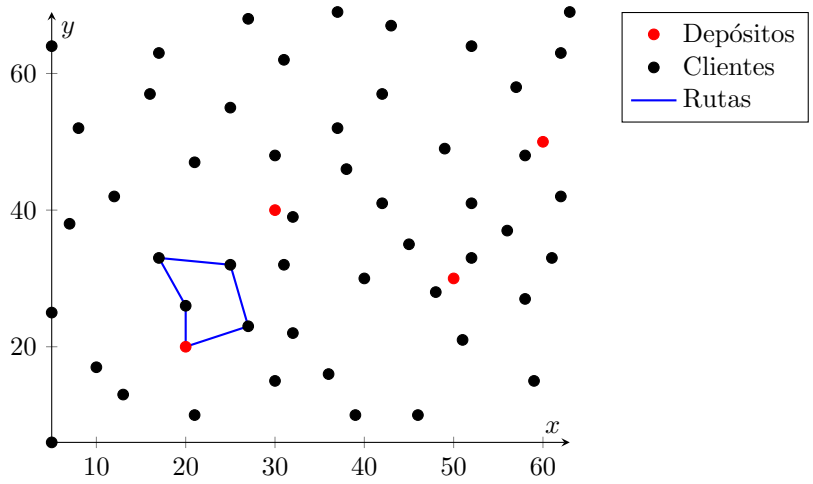

Figura 1: Representación de Depósitos, Clientes y Rutas de la instancia p01

Este esquema proporciona una visión clara de cómo se organiza y se visualiza la solución del problema, mostrando claramente la interacción entre depósitos, clientes y rutas planificadas. Además, la ruta que se observa en el esquema es una **solución generada por el algoritmo Greedy**.

6. Descripción del algoritmo

La solución implementada para el problema de rutas de vehículos combina un enfoque Greedy para la inicialización de la solución y una búsqueda Tabú (*Tabu Search*) para su mejora. El objetivo es optimizar la asignación de rutas a los vehículos para minimizar la distancia total recorrida, respetando las restricciones de capacidad de los vehículos y las distancias máximas

permitidas por los depósitos. El algoritmo se estructura en dos fases principales: una fase de inicialización utilizando un enfoque Greedy y una fase de mejora mediante la búsqueda Tabú.

6.1. Fase de Inicialización: Algoritmo Greedy

La fase de inicialización se basa en una algoritmo Greedy, que genera una solución de manera rápida y eficiente. Este algoritmo selecciona iterativamente el cliente más cercano que cumpla con las restricciones de capacidad del vehículo y la distancia máxima desde el depósito. La inicialización de las rutas comienza creando **M * T** rutas, donde **M** es el número de vehículos por depósito y **T** es el número de depósitos. Cada ruta se inicializa con el **ID** del depósito correspondiente.

Algorithm 1 Algoritmo Greedy

```
1:  function GREEDY(M, N, T, depositos, clientes)
2:      rutas ← vector < Ruta > (M * T)
3:      for i ← 0 to (M * T) − 1 do
4:          rutas[i].depositoId ← (i/M) + 1
5:      end for
6:      clienteVisitado ← vector < bool > (N, false)
7:      for each ruta in rutas do
8:          depositoActual ← coordenadas del depósito correspondiente
9:          while true do
10:             clienteMasCercano ← −1
11:             distanciaMinima ← ∞
12:             for j ← 0 to N − 1 do
13:                 if !clienteVisitado[j] and cumple restricciones then
14:                     distancia ← calcular distancia entre clienteActual y cliente[j]
15:                     if distancia < distanciaMinima and no viola restricciones then
16:                         distanciaMinima ← distancia
17:                         clienteMasCercano ← j
18:                     end if
19:                 end if
20:             end for
21:             if clienteMasCercano = −1 then
22:                 if clienteActual ≠ −1 then
23:                     ruta.distanciaRecorrida ← ruta.distanciaRecorrida +
24:                       distancia(clienteActual, depósito)
25:                 end if
26:                 break
27:             end if
28:             clienteVisitado[clienteMasCercano] ← true
29:             ruta.clientesVisitados.push_back(clienteMasCercano + 1)
30:             ruta.cargaActual ← ruta.cargaActual + clientes[clienteMasCercano].demanda
31:             ruta.distanciaRecorrida ← ruta.distanciaRecorrida + distanciaMinima
32:             clienteActual ← clienteMasCercano
33:         end while
34:     end for
35:     return rutas
36: end function
```

El proceso de asignación de clientes consiste en iterar sobre cada ruta y seleccionar el cliente más cercano que cumpla con las restricciones de capacidad y distancia. Después de cada asigna-

ción, se actualizan las métricas de carga y distancia recorrida por la ruta actual. Si no se puede asignar más clientes a una ruta sin violar las restricciones, se finaliza la ruta y se procede a la siguiente. Este enfoque asegura una solución inicial factible que sirve como punto de partida para la búsqueda Tabú.

6.2. Fase de Mejora: Búsqueda Tabú

Para mejorar la solución inicial obtenida por el algoritmo Greedy, se emplea la búsqueda Tabú. Este método explora el espacio de soluciones mediante movimientos de intercambio (*swap*) de clientes entre rutas, evitando ciclos mediante el uso de una lista Tabú. La búsqueda Tabú comienza con la inicialización de parámetros, definiendo el tamaño de la lista Tabú, número de máximo de iteraciones y la lista tabú vacía.

Algorithm 2 Tabu Search

```
1: function TABUSEARCH(rutas, clientes, depositos, gen, tabuTenure, maxIteraciones)
2:     tabuList ← unordered_set < string > ()
3:     mejorSolucion ← rutas
4:     mejorDistancia ← evaluateSolution(rutas)
5:     iteracion ← 0, iteracionSinMejora ← 0
6:     maxIteracionesSinMejora ← 10
7:     while iteracion < maxIteraciones and
8:             iteracionSinMejora < maxIteracionesSinMejora do
9:         if swapMovement(rutas, clientes, depositos, gen, tabuList) then
10:            nuevaDistancia ← evaluateSolution(rutas)
11:            if nuevaDistancia < mejorDistancia then
12:                mejorDistancia ← nuevaDistancia
13:                mejorSolucion ← rutas
14:                tabuList.clear(), iteracionSinMejora ← 0
15:            else
16:                iteracionSinMejora ← iteracionSinMejora + 1
17:            end if
18:        else
19:            iteracionSinMejora ← iteracionSinMejora + 1
20:        end if
21:        if tabuList.size() ≥ tabuTenure then
22:            tabuList.erase(tabuList.begin())
23:        end if
24:        iteracion ← iteracion + 1
25:    end while
26:    return mejorSolucion
27: end function
```

La evaluación de la solución se realiza calculando la distancia total recorrida de la solución inicial, que se utiliza como referencia. En el bucle principal de la búsqueda, se realiza un movimiento de intercambio seleccionando aleatoriamente dos clientes y cambiándolos entre rutas (o dentro de la mis ruta). Se verifica que el intercambio no viole las restricciones de capacidad y distancia. Si un movimiento mejora la solución conocida, se permite incluso si está en la lista Tabú (criterio de aspiración). Después de cada movimiento, se actualiza la lista Tabú para evitar ciclos.

El movimiento de intercambio (*swap*) se selecciona por su capacidad para explorar eficazmente el espacio de soluciones. Este operador permite reordenar clientes entre rutas de manera

flexible, potenciando tanto la intensificación (mejora de soluciones locales) como la diversificación (exploración de nuevas soluciones). La implementación de un criterio de aspiración garantiza que se puedan superar barreras locales si se encuentran una mejora significativa, manteniendo el balance entre la búsqueda local y la exploración global.

Por último, se tiene que las estructuras utilizadas incluyen definiciones para almacenar los atributos de los clientes y depósitos, y una estructura para las rutas que almacenan la información de cada ruta, incluyendo los clientes visitados, la carga actual y la distancia recorrida. La función de evaluación calcula la distancia total recorrida por todas las rutas, considerando tanto las distancias entre clientes como las distancias de retorno al depósito.

7. Experimentos

Esta sección presenta experimentos destinados a evaluar y optimizar el algoritmo *Tabu Search* en la resolución del problema de rutas de vehículos. Se exploran tres aspectos claves: la longitud de la lista tabú, la variabilidad de la semilla aleatoria y el impacto de intercambios aleatorios antes de aplicar el algoritmo Greedy. Cada experimento se enfoca en comprender cómo estos factores afectan la calidad de las soluciones y la eficiencia computacional del algoritmo.

7.1. Experimento 1: Variación de la longitud de la lista tabú

El primer experimento investiga cómo varía el desempeño del algoritmo *Tabu Search* al modificar la longitud de la lista tabú. Se seleccionan cinco instancias diferentes de *MDVRP* (p01, p02, p03, p04, p05), cada una con múltiples depósitos y clientes de diferentes tamaños y complejidades. Las ejecuciones del algoritmo se realizan con listas tabú de longitud 2 y 5 movimientos prohibidos, además del promedio de la cantidad de clientes por ruta, siendo esta cantidad los movimientos prohibidos. Los criterios de término incluyen un máximo de 1000 iteraciones o un estancamiento durante 50 iteraciones consecutivas. Se evalúan la distancia total recorrida y el tiempo de ejecución para determinar la configuración óptima de la lista tabú.

Además, se sabe que la longitud de la lista tabú influye directamente en la exploración del espacio de soluciones y en la capacidad del algoritmo para escapar de mínimos locales. Variar esta longitud permite encontrar un equilibrio entre exploración y explotación.

7.2. Experimento 2: Modificación de la Semilla Aleatoria

El segundo experimento explora cómo la variación de la semilla aleatoria afecta el rendimiento del algoritmo *Tabu Search* en las mismas instancias de *MDVRP* usadas en el Experimento 1 (7.1). Se ejecutan múltiples iteraciones del algoritmo con tres semillas diferentes para cada instancia de prueba. Se registran la distancia total recorrida y el tiempo de ejecución para analizar la consistencia y robustez del algoritmo ante cambios en las condiciones iniciales.

La semilla afecta la exploración del espacio de búsqueda. Variar la semilla permite evaluar la estabilidad y reproducibilidad del algoritmo frente a diferentes condiciones iniciales.

7.3. Experimento 3: Intercambio de clientes antes de aplicar Greedy

El tercer experimento evalúa el impacto de realizar intercambios aleatorios entre clientes antes de aplicar el algoritmo *Greedy* para la construcción inicial de rutas. Se seleccionan 5 instancias de *MDVRP* con diversas características. Se lleva a cabo un proceso de preparación donde se intercambian aleatoriamente la mitad de la cantidad de clientes, otra preparación fue intercambiando aleatoriamente la cantidad de clientes y en último lugar cincuenta intercambios aleatorios (como valor estático).

Los intercambios aleatorios pueden ayudar a evitar soluciones sub-óptimas en la fase inicial de construcción de rutas. Este experimento busca evaluar si mejorar la diversidad inicial de las soluciones influye en la calidad de soluciones finales.

8. Resultados

Esta sección presenta los resultados obtenidos a partir de los experimentos realizados para evaluar y optimizar el algoritmo *Tabu Search* en la resolución de *MDVRP*. Se analizan tres aspectos: longitud de la lista tabú, la variabilidad de la semilla aleatoria y el impacto de intercambios aleatorios antes de aplicar el algoritmo Greedy.

8.1. Resultados Experimento 1: Variación de la longitud de la lista tabú

En el primer experimento se evaluó el desempeño del algoritmo *Tabu Search* al variar la longitud de la lista tabú.

Instancia	**Longitud Lista Tabú**	**Tiempo de Ejecución [s]**	**Z**
p01	2	8.7e-05	1043.33
	5	8.7e-05	1043.33
	Promedio clientes/ruta	8.7e-05	1043.33
p02	2	0.000122	772.69
	5	0.000133	772.69
	Promedio clientes/ruta	0.000126	772.69
p03	2	0.000114	972.91
	5	0.000115	972.91
	Promedio clientes/ruta	0.000147	972.91
p04	2	0.000107	1717.09
	5	0.000111	1717.09
	Promedio clientes/ruta	0.000108	1717.09
p05	2	0.000128	1234.91
	5	0.000127	1234.91
	Promedio clientes/ruta	0.000129	1234.91

Cuadro 1: Resultados de la variación de la longitud de la lista tabú

Se logra observar que los tiempos de ejecución fueron muy similares entre las diferentes longitudes de la lista tabú, lo que indica que la lista no afecta significativamente el tiempo de procesamiento. Por otro lado, la calidad de la solución (Z), medida en términos de la distancia total recorrida, fue la misma para todas las longitudes de la lista tabú. Esto sugiere que para las instancias consideradas, la longitud de la lista tabú no tiene un impacto significativo en la calidad de las soluciones obtenidas.

8.2. Resultados Experimento 2: Modificación de la Semilla Aleatoria

En el segundo experimento se investigó cómo la variabilidad de la semilla aleatoria afecta el rendimiento del algoritmo *Tabu Search*.

Instancia	Semilla	Tiempo de Ejecución [s]	Z
p01	10	0.000101	1083.09
	100	9.1e-05	1016.84
	1000	9.7e-05	1001.83
p02	10	0.000124	770.95
	100	0.000126	743.90
	1000	0.000124	772.69
p03	10	0.000103	1029.21
	100	0.000114	975.84
	1000	0.000114	1029.21
p04	10	0.000103	1619.40
	100	0.000103	1742.63
	1000	0.000121	1611.30
p05	10	0.000140	1232.56
	100	0.000126	1199.15
	1000	0.000130	1199.15

Cuadro 2: Resultados de la modificación de la Semilla Aleatoria

Se observa que los tiempos de ejecución mostraron pequeñas variaciones entre las diferentes semillas, pero en general fueron consistentes. Por otro lado, la calidad de la solución (Z), sugiere que la semilla aleatoria tiene un impacto considerable en la calidad de las soluciones. Esto destaca la importancia de realizar múltiples ejecuciones con diferentes semillas para obtener soluciones robustas y de alta calidad.

8.3. Resultados Experimento 3: Intercambio de clientes antes de aplicar Greedy

En el tercer experimento se evaluó el impacto de realizar intercambios aleatorios entre clientes antes de aplicar el algoritmo Greedy.

Instancia	Cantidad de Swaps	Tiempo de Ejecución [s]	Z
p01	Largo/2	8.7e-05	1043.33
	Largo	8.8e-05	1001.83
	50	9.0e-05	1001.83
p02	Largo/2	0.000182	772.69
	Largo	0.000133	755.11
	50	0.000138	755.11
p03	Largo/2	0.000118	972.91
	Largo	0.000108	936.42
	50	0.000108	975.84
p04	Largo/2	0.000105	1717.09
	Largo	0.000187	1755.49
	50	0.000104	1717.09
p05	Largo/2	0.000136	1234.91
	Largo	0.000131	1234.91
	50	0.000130	1234.91

Cuadro 3: Resultados del intercambio de clientes antes de aplicar Greedy

Se observa que el tiempo de ejecución mostró variaciones entre las diferentes cantidades de intercambios, siendo generalmente similares en todos los casos. Por otro lado, la calidad de la

solución (Z), indican que realizar intercambios aleatorios puede mejorar significativamente la calidad de las soluciones iniciales, especialmente cuando se realizan intercambios equivalente al largo total de los clientes. Esto sugiere que una mayor diversidad inicial de las soluciones conduce a mejores resultados finales después de aplicar *Tabu Search*.

9. Conclusiones

El *Multi-Depot Vehicle Routing Problem* es de suma relevancia en el ámbito logístico, dado que su resolución eficiente puede minimizar costos operativos y mejorar la eficiencia en la distribución. Este estudio ha evaluado diversas técnicas para abordar el *MDVRP*, enfocándose particularmente en el algoritmo *Tabu Search* y su comportamiento bajo diferentes parámetros de experimentación.

Todas las técnicas revisadas, basadas en programación lineal, comparten el objetivo de optimizar las rutas de vehículos desde múltiples depósitos hacia los clientes. Estas técnicas buscan minimizar costos y maximizar la eficiencia, apoyándose en heurísticas y algoritmos de búsqueda para encontrar soluciones factibles en un tiempo razonable, dada la complejidad computacional del *MDVRP*.

Los experimentos realizados proporcionaron información valiosa sobre el comportamiento del algoritmo *Tabu Search*. En primer lugar, se concluyó que la longitud de la lista tabú no afecta significativamente ni el tiempo de ejecución ni la calidad de las soluciones para las instancias estudiadas. Esto sugiere que, al menos para los problemas y parámetros específicos considerados, la longitud de la lista puede ser un factor menos crítico en comparación con otros parámetros.

En segundo lugar, se observó que la semilla aleatoria tiene un impacto considerable en la calidad de las soluciones, lo que subraya la importancia de realizar múltiples ejecuciones con diferentes semillas para obtener soluciones robustas y de alta calidad. La variabilidad de la semilla puede llevar a diferentes soluciones, algunas de las cuales pueden ser significativamente mejores que otras.

Finalmente, los resultados indicaron que realizar intercambios aleatorios de clientes antes de aplicar el algoritmo Greedy puede mejorar significativamente la calidad de las soluciones iniciales. Esto sugiere que una mayor diversidad inicial de las soluciones conduce a mejores resultados finales después de aplicar *Tabu Search*.

Entre las ventajas de la propuesta se destaca su flexibilidad, ya que el algoritmo *Tabu Search* puede ajustarse a diferentes configuraciones de problemas y parámetros. La posibilidad de realizar múltiples ejecuciones con diferentes semillas aleatorias permite explorar un mayor espacio de soluciones, lo que puede resultar en la identificación de rutas de alta calidad. Además, los intercambios aleatorios antes del proceso Greedy mejoran la diversidad inicial, lo que puede llevar a mejores soluciones finales.

Sin embargo, también se identifican algunas desventajas. La calidad de las soluciones puede ser altamente dependiente de la configuración de parámetros, como la semilla aleatoria, lo que puede requerir ajustes finos para diferentes instancias del problema. Aunque el tiempo de ejecución fue relativamente consistente, en problemas de mayor tamaño o complejidad, el tiempo de computación puede aumentar significativamente.

Para futuras investigaciones, se sugieren varias áreas de desarrollo. La exploración de enfoques alternativos, como variantes avanzadas de algoritmos genéticos, podría mejorar la calidad de las soluciones obtenidas. Además, desarrollar técnicas de optimización que consideren múltiples objetivos simultáneamente, como minimizar costos y tiempo de entrega, puede proporcionar soluciones más integrales.

La incorporación de técnicas de aprendizaje automático y análisis predictivo podría mejorar la precisión de los modelos y la adaptabilidad de las soluciones en tiempo real. Considerar factores dinámicos, como tráfico y condiciones climáticas, mediante la integración de datos en tiempo real, también puede mejorar la robustez de los enfoques existentes.

Referencias

[1] H. Q. Nguyen B. L. Golden, T. L. Magnanti. Implementing vehicle routing algorithms. *Networks*, 7(2):113–148, 1977.

[2] F.T. Hanshar B. Ombuki-Berman. Using genetic algorithms for multi-depot vehicle routing. *Studies in Computational Intelligence*, 161:77–99, 2009.

[3] J.O. Cerdeira. On the multi-depot vehicle routing problem. *Operations Research '91*, pages 144–147, 1992.

[4] J. H. Ramser G. B. Dantzig. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[5] S. Taillefer G. Laporte, Y. Nobert. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science*, 22(3):161–172, 1988.

[6] I.O. Viera I.D. Giosa, I.L. Tansini. New assignment algorithms for the multi-depot vehicle routing problem. *Journal of the Operational Research Society*, 53(9):977–984, 2002.

[7] F.F. Boctor J. Renaud, G. Laporte. An improved petal heuristic for the vehicle routing problem. *Journal of the Operational Research Society*, 47(2):329–336, 1996.

[8] F.F. Boctor J. Renaud, G. Laporte. A tabu search heuristic for the multi-mepot vehicle routing problem. *Computers & Operations Research*, 23(3):229–235, 1996.

[9] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

[10] H.P. Crowder P. Wolfe. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.

[11] A.P.B. Póvoa T.R.P. Ramos, M.I. Gomes. Multi-depot vehicle routing problem: a comparative study of alternative formulations. *International Journal of Logistics Research and Applications*, 23(2):103–120, 2020.