**Optimización de rutas para problema de Enrutamiento de Vehículos Abiertos (OVRP): Un enfoque matemático y de solución**

Diego Gasset, Felipe Paillalef, Sebastián Payacán

Facultad de Ingeniería, Universidad Andrés Bello, Santiago, Chile, 2023

**ABSTRACT:** En el constante panorama de movilidad humana y comercio, la planificación eficiente de rutas se ha vuelto primordial. Este artículo aborda el Problema de Ruteo de Vehículos Abierto (OVRP), un desafío logístico significativo en la optimización de rutas para una flota de vehículos que atiende a clientes geográficamente dispersos. Utilizando un enfoque heurístico, exploramos las complejidades del OVRP, comparando resultados con métodos de vanguardia como LKH-3 y AMPL. Nuestro estudio no solo destaca la efectividad de la modelización matemática, sino que también explora la practicidad de algoritmos heurísticos como Greedy, Vecino más Cercano y 2-opt para ofrecer soluciones de calidad. Los hallazgos subrayan la interacción matizada entre la calidad de la solución y la eficiencia computacional, proporcionando perspicacias valiosas para abordar desafíos logísticos del mundo real. Además, nuestras recomendaciones profundizan en oportunidades de optimización y la integración de tecnologías emergentes, asegurando soluciones adaptables para el intrincado Problema de Ruteo de Vehículos Abierto.

*Keywords: Mathematical model, Heterogeneous fleet, Single depot, Open vehicle routing problem*

# INTRODUCTION

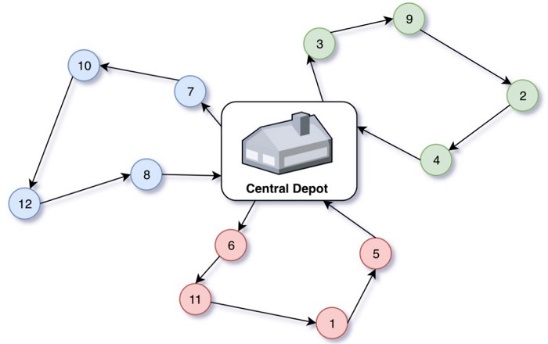
La necesidad de desplazarse de un punto a otro ha sido una constante en la evolución humana durante miles de años. Aunque los usos e importancia han aumentado significativamente con el tiempo, el objetivo fundamental de facilitar el movimiento persiste.

Este impulso por la movilidad ha sido crucial en el ámbito comercial a lo largo de la historia. Desde las antiguas rutas comerciales como la Ruta de la Seda, que conectaba Oriente y Occidente a través de Asia, Europa y África, hasta el presente, el comercio y la entrega de productos han experimentado una evolución notable, respaldada por avances significativos en la planificación de rutas.

La planificación de rutas se ha convertido en una prioridad para reducir los tiempos y las distancias de recorrido en la entrega de productos. Uno de los primeros problemas planteados para abordar esta situación fue el Problema del Vendedor Viajero (Travelling Salesman Problem, TSP), que busca encontrar la ruta más corta para un viajante de comercio que debe visitar un grupo especificado de ciudades y regresar al punto de partida original (Dantzig et al., 1954).

Esta problemática se generalizó para abordar la planificación de rutas para vehículos de logística, dando origen al Problema de Ruteo de Vehículos (Vehicle Routing Problem, VRP). Este problema implica determinar un conjunto de rutas para satisfacer las solicitudes de transporte con la flota de vehículos al costo mínimo y regresar al depósito cumpliendo un ciclo Hamiltoniano, optimizando la ejecución de todas las rutas (Toth and Vigo, 2014).

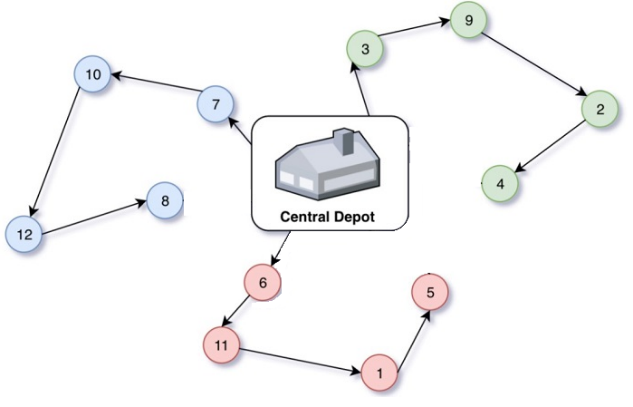
Fig.1 Problema clásico de enrutamiento de vehículos (VRP)



Con el tiempo, algunas empresas han optado por externalizar la logística y el transporte, permitiéndoles centrarse en sus competencias principales y ahorrar costos externalizando estas funciones a proveedores de logística de terceros (Zhen et al., 2020).

Sin embargo, para la planificación de rutas realizada por estos proveedores de logística de terceros, el VRP no es suficiente, ya que los vehículos no necesariamente regresan al depósito al final del recorrido. Para resolver esta complejidad, se emplea el Problema de Ruteo de Vehículos Abierto (Open Vehicle Routing Problem, OVRP), una variante del VRP que tiene en cuenta la capacidad del vehículo y no obliga al retorno al punto de partida, permitiendo que cada ruta termine en uno de los clientes (Brandão, 2014). Otros factores, como la cantidad de vehículos y la demanda por cliente, también se consideran en este problema.

Fig.2 Problema de enrutamiento de vehículos abiertos



Dada la complejidad y relevancia del OVRP en escenarios logísticos reales, abordar sus desafíos requiere enfoques heurísticos y técnicas de optimización avanzadas. Este estudio tiene como objetivo llevar a cabo un análisis profundo de herramientas de optimización específicas para este problema, junto con la implementación de dos heurísticas para su resolución. La comparación de resultados resultará en un aumento del conocimiento heurístico para afrontar de manera más efectiva el OVRP.

# Metodología de investigación

La metodología de investigación consta de 4 partes, estudiar el problema actual de entrega y sus restricciones, formular un modelo matemático en lenguaje AMPL, Comparar resultados con LKH-3, desarrollar algoritmos de solución y finalmente llevar a cabo experimentos.

# Experimentación

En nuestro estudio, que se centra en el problema de ruteo de vehículos (VRP), adoptamos un enfoque multidimensional para la modelización y resolución de este desafío. Inicialmente, utilizamos AMPL (A Mathematical Programming Language), un lenguaje de programación especializado en problemas de programación matemática, para formular y resolver variantes del VRP. Esta herramienta es ideal para abordar una gama de problemas de optimización, desde la programación lineal hasta la programación entera.

Para garantizar resultados consistentes y comparables, todas las pruebas se llevaron a cabo en un entorno de hardware específico, empleando un procesador Intel Core i5 10400F a 2.9 GHz con 32 GB de memoria RAM DDR4 a 3200 MHz. La siguiente tabla presenta los resultados obtenidos a través de AMPL, comparándolos con los alcanzados mediante el uso de LKH-3, una evolución avanzada del algoritmo de Lin-Kernighan-Helsgaun.

LKH-3, aplicado en nuestro experimento, es un algoritmo destacado por su eficiencia y eficacia en la resolución de problemas tanto del Viajante de Comercio (TSP) como del VRP. Este algoritmo se basa en una heurística de búsqueda local que optimiza las rutas mediante la técnica k-opt, con un valor de k que varía dinámicamente. Su flexibilidad para adaptarse a distintas restricciones lo hace una herramienta comparativa ideal para nuestro modelo. Implementamos LKH-3 en una serie de instancias representativas para el benchmarking, evaluando su rendimiento y las soluciones generadas frente a las obtenidas por nuestro algoritmo. Esta comparación no solo valida y calibra nuestra metodología, sino que también abre nuevas perspectivas y aplicaciones prácticas en optimización de rutas y logística.

En el marco de nuestro estudio, definimos las heurísticas como métodos de resolución que aplican estrategias prácticas, basadas en la experiencia y el conocimiento específico del problema, para encontrar soluciones de alta calidad en un tiempo razonable. Dada la complejidad y la naturaleza a menudo NP-difícil del VRP, especialmente en sus variantes con múltiples restricciones y objetivos, las heurísticas como LKH-3 son fundamentales. Estos métodos manejan eficientemente la amplia búsqueda de espacios de soluciones y las complejas interacciones entre variables y restricciones del VRP, tales como capacidades de vehículos, ventanas de tiempo y rutas de retorno.

# 3.1 Diseño de experimento

La experimentación computacional propuesta tiene como propósito la comparación entre las herramientas AMPL, LKH-3 y heurísticas implementadas en Python, en donde será evaluando su desempeño en términos de funcionalidad, tiempo de ejecución y resultados en la resolución del problema. Dentro de la fase inicial, se procederá a la definición precisa del problema y al fin de la experimentación que se quiere lograr, con el fin de poder ir estableciendo objetivos y criterios de rendimiento con claridad. Subsecuentemente, se generarán códigos para la aplicación de heurísticas, para ser usados dentro de Python, con un enfoque particular en la eficiencia y optimización de dichas implementaciones. Paralelamente, se configurarán las herramientas AMPL y LKH-3 para su uso dentro de la experimentación. La fase de diseño experimental comprenderá la generación de instancias de prueba que abarquen diversas características del problema, junto con la instauración de una metodología de ejecución pormenorizada para cada herramienta. La experimentación en sí abarcará la ejecución de cada herramienta en las instancias de prueba, registrando meticulosamente el tiempo de ejecución, la calidad de los resultados obtenidos, además de otras métricas pertinentes. Se llevará a cabo un análisis estadístico de los resultados con el fin de validar cualquier diferencia significativa observada. El informe final será exhaustivo en su documentación, incorporando detalles inherentes a la experimentación, visualizaciones gráficas y tablas con el propósito de facilitar la comprensión de los resultados. Asimismo, se contemplará un proceso iterativo de revisión y mejora continua, evaluando posibles ajustes en las implementaciones y la metodología experimental para perfeccionar la comparación entre las herramientas. Los recursos necesarios para la realización de este experimento incluirán hardware con especificaciones adecuadas y software especializado, AMPL, LKH-3 y Python, para la ejecución del estudio.

El enfoque de modelaje adoptado para este proyecto se fundamentará en la adaptación del Vehicle Routing Problem (VRP) propuesto por Toth y Tigo, específicamente modificado para abordar las características únicas del Open Vehicle Routing Problem (OVRP). En la formulación de nuestro modelo, nos inspiramos en la capacidad de carga y la restricción de no regresar al punto de origen, lo cual representa una variante significativa con respecto al VRP tradicional.

Para la evaluación y comparación de los resultados obtenidos en este estudio, se emplearán instancias provenientes de LKH-3. Este conjunto se seleccionó debido a la reconocida eficacia de LKH-3 en la resolución del problema OVRP mediante la aplicación de heurísticas. Al formular el problema de manera matemática para su implementación en AMPL, se ha establecido una restricción temporal que limita la ejecución a un máximo de 1 hora, considerando la naturaleza continua y la necesidad de configuración detallada en comparación con códigos abiertos en Python. En particular, se utilizará el solver Cplex en AMPL para la obtención de las soluciones.

Posteriormente, se procederá a la presentación y comparación de los resultados obtenidos en una tabla que incluirá las soluciones proporcionadas por LKH-3, el Best Known Solution (BKS) como referencia principal, la aplicación de nuestro modelo dentro de AMPL con la restricción temporal mencionada, y la implementación de heurísticas desarrolladas en Python. Esta estructuración permitirá una evaluación integral de la eficacia de cada enfoque, destacando tanto la calidad de las soluciones como la eficiencia temporal de los algoritmos empleados.

### Análisis de Datos

Esta tabla estructurada proporciona una visión detallada de los resultados obtenidos para cada instancia, permitiendo la comparación de los diferentes métodos en función de parámetros clave del problema OVRP. Además, se incluye el tiempo correspondiente de ejecución (T') para evaluar la eficiencia temporal de cada enfoque.

Dentro de la presente tabla se encuentra:

* **Instancia**
* **n** (Numero de Nodos dentro de la instancia)
* k (La cantidad de Vehículos utilizados en la instancia)
* **LKH-3**
* **AMPL**
* **H.1** (Greedy)
* **H.2** (Nearest Neighbor)
* **H.3** (Local Search)
* **H.4** (Nearest Neighbor+ Local Search)
* T' (El respectivo tiempo de cada aplicación)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instancia | n | k | LKH-3 | AMPL | T’ | H1 | T’ | H2 | T’ | H3 | T’ | H4 | T’ |
| A-n45-k7 | 45 | 7 | 685.16 | 1223.20 | 1 Hr | 828.23 | 0.003 S | 828.23 | 0.01 S | 1469.48 | 0.08S | 801.16 | 0.02S |
| A-n46-k7 | 46 | 7 | 583.54 | 934.44 | 1 Hr | 730.42 | 0.003 S | 730.42 | 0.01 S | 1533.13 | 0.01S | 711.32 | 0.02S |
| A-n48-k7 | 48 | 7 | 669.83 | 1215.71 | 1 Hr | 824.81 | 0.003 S | 824.81 | 0.01S | 1510.12 | 0.01S | 824.81 | 0.02S |
| A-n53-k7 | 53 | 7 | 655.18 | 1086.32 | 1 Hr | 742.07 | 0.004 S | 742.07 | 0.01S | 1785.88 | 0.01S | 719.77 | 0.02 S |
| A-n54-k7 | 54 | 7 | 709.27 | 1259.88 | 1 Hr | 801.21 | 0.005 S | 801.21 | 0.01S | 1720.43 | 0.01S | 790.05 | 0.02 S |
| A-n55-k9 | 55 | 9 | 669.06 | 1202.63 | 1 Hr | 781.93 | 0.007 S | 781.93 | 0.01 S | 2016.93 | 0.01S | 774.96 | 0.03 S |
| A-n60-k9 | 60 | 9 | 798.01 | 1497.40 | 1 Hr | 920.00 | 0.005 S | 920.00 | 0.01 S | 2041.83 | 0.009S | 910.78 | 0.02 S |
| A-n61-k9 | 61 | 9 | 678.30 | 1158.38 | 1 Hr | 670.07 | 0,005 S | 670.07 | 0.01 S | 1809.19 | 0.01S | 643.23 | 0.02 S |
| A-n62-k8 | 62 | 8 | 783.18 | 1616.70 | 1 Hr | 998.68 | 0.005 S | 998.68 | 0.01 S | 2041.31 | 0.01S | 995.14 | 0.02 S |
| A-n63-k9 | 63 | 9 | 941.53 | 1948.95 | 1 Hr | 1021.69 | 0.005 S | 1021.69 | 0.01 S | 2069.84 | 0.01S | 1021.69 | 0.03 S |
| A-n63-k10 | 63 | 10 | 778.46 | 1455.23 | 1 Hr | 929.40 | 0.009 S | 929.40 | 0.01 S | 2090.68 | 0.009S | 919.98 | 0..02S |
| A-n64-k9 | 64 | 9 | 848.16 | 1794.15 | 1 Hr | 878.82 | 0.005 S | 878.82 | 0.01 S | 1932.48 | 0.009S | 857.51 | 0.02S |
| A-n65-k9  (Revisar) | 65 | 9 | 728.59 | 1574.76 | 1 Hr | 784.56 | 0.02 S | 784.56 | 0.01S | 2053.60 | 0.009S | 768.80 | 0.02S |
| A-n69-k9 | 69 | 9 | 757.76 | 1335.82 | 1 Hr | 995.80 | 0.005 S | 995.80 | 0.01S | 2134.03 | 0.01S | 924.17 | 0.02 S |
| A-n80-k10  (Revisar) | 80 | 10 | 1067.09 | 2423.15 | 1 Hr | 1199.40 | 0.005 S | 1199.40 | 0.02S | 2560.59 | 0.01S | 1185.16 | 0.03S |
| B-n63-k10  (Revisar) | 63 | 10 | 837.07 | 1677.04 | 1 Hr | 965.74 | 0.004 S | 965.74 | 0.01S | 2254.58 | 0.01S | 943.92 | 0.02S |
| B-n64-k9  (Revisar) | 64 | 9 | 520.47 | 945.12 | 1 Hr | 552.74 | 0.005 S | 552.74 | 0.01S | 1708.14 | 0.01S | 551.33 | 0.02S |
| B-n66-k9  (Revisar) | 66 | 9 | 755.27 | 1514.61 | 1 Hr | 778.79 | 0.006 S | 778.79 | 0.02S | 1810.74 | 0.01S | 771.73 | 0.03S |
| B-n67-k10 | 67 | 10 | 616.54 | 1351.02 | 1 Hr | 716.18 | 0.006 S | 716.18 | 0.01S | 2167.93 | 0.01S | 704.91 | 0.02S |
| B-n68-k9  (Revisar) | 68 | 9 | 701.72 | 1474.59 | 1 Hr | 760.09 | 0.006 S | 760.09 | 0.01S | 1808.70 | 0.01S | 757.67 | 0.02S |
| B-n78-k10  (Revisar) | 78 | 10 | 722.71 | 1573.77 | 1 Hr | 720.99 | 0.008 S | 720.99 | 0.01S | 2147.56 | 0.01S | 717.51 | 0.02S |
| C1  (Revisar) | 51 | 5 | 416.06 | 600.94 | 1 Hr | 515.62 | 0.003 S | 515.62 | 0.01S | 927.84 | 0.01S | 491.72 | 0.02S |
| C2  (Revisar) | 76 | 10 | 567.14 | 997.23 | 1 Hr | 607.42 | 0.008 S | 607.42 | 0.02S | 1493.14 | 0.01S | 602.16 | 0.02S |
| C3 | 101 | 8 | 639.74 | 1003.35 | 1 Hr | 817.53 | 0.01 S | 817.53 | 0.02S | 1606.92 | 0.01S | 783.92 | 0.03S |
| C4 | 151 | 12 | 733.13 | 1443.91 | 1 Hr | 970.36 | 0.03 S | 970.36 | 0.02S | 2409.30 | 0.02S | 955.43 | 0.03S |
| C5 | 200 | 17 | 879.43 | 2380.71 | 1 Hr | 1112.43 | 0.05 S | 1112.45 | 0.02S | 3300.45 | 0.03S | 1086.41 | 0.04S |
| C11 | 121 | 7 | 682.12 | 1421.53 | 1 Hr | 725.18 | 0.01 S | 725.18 | 0.02S | 2106.23 | 0.02S | 705.19 | 0.03S |
| C12 | 101 | 10 | 534.24 | 1110.40 | 1 Hr | 756.02 | 0.01 S | 756.02 | 0.02S | 2211.76 | 0.01S | 731.86 | 0.03S |
| E-n23-k3  (Revisar) | 23 | 3 | 442.98 | 568.52 | 161 S | 385.38 | 0.01 S | 385.38 | 0.01S | 731.05 | 0.01S | 380.25 | 0.02S |
| E-n30-k3 | 30 | 3 | 393.51 | 505.01 | 1 Hr | 475.20 | 0.001 S | 475.20 | 0.01S | 627.63 | 0.01S | 463.56 | 0.02S |
| E-n51-k5  (Revisar) | 51 | 5 | 416.06 | 606.07 | 1 Hr | 515.62 | 0.002 S | 515.62 | 0.03S | 927.84 | 0.009S | 491.72 | 0.04S |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E-n76-k7 | 76 | 7 | 530.02 | 845.04 | 1 Hr | 734.20 | 0.007 s | 734.20 | 0.01S | 1244.84 | 0.01S | 667.25 | 0.02S |
| E-n76-k8 | 76 | 8 | 537.24 | 982.66 | 1 Hr | 737.56 | 0.006 s | 737.56 | 0.01S | 1277.75 | 0.01S | 711.36 | 0.03S |
| E-n76-k10  (Revisar) | 76 | 10 | 567.14 | 997.23 | 1 Hr | 607.42 | 0.007 S | 607.42 | 0.02S | 1493.14 | 0.01S | 602.16 | 0.03S |
| E-n76-k14  (Revisar) | 76 | 14 | 623.55 | 1199.05 | 1 Hr | 621.07 | 0.008 S | 621.07 | 0.02S | 1728.74 | 0.009S | 621.07 | 0.03S |
| E-n101-k8 | 101 | 8 | 639.74 | 1010.32 | 1 Hr | 817.53 | 0.01 S | 817.53 | 0.01S | 1606.92 | 0.01S | 783.92 | 0.03S |
| E-n101-k14  (Revisar) | 101 | 14 | 711.58 | 1301.88 | 1 Hr | 866.69 | 0.01 S | 866.69 | 0.02S | 2168.04 | 0.01S | 844.92 | 0.03S |
| F10 | 45 | 4 | 463.90 | 727.74 | 1 Hr | 643.15 | 0.003 s | 643.15 | 0.01S | 1372.94 | 0.01S | 587.63 | 0.02S |
| F11  (Revisar) | 72 | 4 | 177.00 | 333.06 | 1 Hr | 174.63 | 0.006 s | 174.63 | 0.01S | 465.76 | 0.02S | 163.26 | 0.03S |
| F12 | 135 | 7 | 769.55 | 1501.52 | 1 Hr | 924.12 | 0.02 S | 924.12 | 0.02S | 2659.48 | 0.03S | 884.81 | 0.04S |
| M-n101-k10 | 101 | 10 | 534.24 | 1111.53 | 1 Hr | 756.02 | 0.01 S | 756.02 | 0.01S | 2211.76 | 0.01S | 731.86 | 0.03S |
| M-n121-k7  (Revisar) | 121 | 7 | 682.12 | 1443.95 | 1 Hr | 725.18 | 0.01 S | 725.18 | 0.02S | 2106.23 | 0.02S | 705.19 | 0.04S |
| M-n151-k12 | 151 | 12 | 733.13 | 1443.91 | 1 Hr | 970.36 | 0.03 S | 970.36 | 0.02S | 2409.30 | 0.01S | 955.43 | 0.03S |
| M-n200-k16 (Rev) | 200 | 16 | 892.65 | 2370.78 | 1 Hr | 1024.78 | 0.05 S | 1024.78 | 0.02S | 3206.44 | 0.02S | 998.75 | 0.04S |
| M-n200-k17 | 200 | 17 | 867.89 | 2380.71 | 1 Hr | 1112.45 | 0.05 S | 1112.45 | 0.04S | 3300.45 | 0.02S | 1086.41 | 0.05S |
| P-n50-k8  (Revisar) | 50 | 8 | 436.51 | 663.73 | 1 Hr | 396.51 | 0.003 S | 396.51 | 0.01S | 940.39 | 0.01S | 396.51 | 0.02S |
| P-n50-k10  (Revisar) | 50 | 10 | 440.44 | 764.01 | 1 Hr | 460.37 | 0.003 S | 460.37 | 0.01S | 1051.33 | 0.01S | 460.37 | 0.02S |
| P-n51-k10  (Revisar) | 51 | 10 | 480.78 | 778.13 | 1 Hr | 470.29 | 0.003 S | 470.29 | 0.01S | 1193.21 | 0.01S | 457.88 | 0.02S |
| P-n55-k7 | 55 | 7 | 411.58 | 595.83 | 1 Hr | 521.06 | 0.004 S | 521.06 | 0.01S | 1101.21 | 0.01S | 509.63 | 0.03S |
| P-n55-k8 | 55 | 8 | 412.55 | 667.90 | 1 Hr | 541.82 | 0.004 S | 541.82 | 0.05S | 1047.54 | 0.01S | 501.58 | 0.07S |
| P-n55-k10 | 55 | 10 | 444.31 | 749.12 | 1 Hr | 584.36 | 0.004 S | 584.36 | 0.01S | 1170.59 | 0.01S | 558.18 | 0.02S |
| P-n60-k10  (Revisar) | 60 | 10 | 482.09 | 872.29 | 1 Hr | 550.46 | 0.004 S | 550.46 | 0.01S | 1196.87 | 0.01S | 541.28 | 0.02S |
| P-n60-k15  (Revisar) | 60 | 15 | 569.43 | 1097.64 | 1 Hr | 620.50 | 0.005 S | 620.50 | 0.01S | 1524.47 | 0.01S | 603.02 | 0.02S |
| P-n65-k10 | 65 | 10 | 522.50 | 887.40 | 1 Hr | 698.51 | 0.005 S | 698.51 | 0.01S | 1316.29 | 0.01S | 664.71 | 0.02S |
| P-n70-k10  (Revisar) | 70 | 10 | 552.65 | 978.38 | 1 Hr | 630.59 | 0.006 S | 630.59 | 0.01S | 1417.13 | 0.01S | 630.59 | 0.05S |
| P-n76-k4  (Revisar) | 76 | 4 | 522.95 | 698.68 | 1 Hr | 579.43 | 0.008 S | 579.43 | 0.01S | 1059.51 | 0.01S | 567.80 | 0.03S |
| P-n76-k5 | 76 | 5 | 525.64 | 746.68 | 1 Hr | 654.19 | 0.008 S | 654.19 | 0.01S | 1032.47 | 0.01S | 631.05 | 0.03S |
| P-n101-k4 | 101 | 4 | 621.75 | 812.47 | 1 Hr | 773.54 | 0.01 S | 773.54 | 0.02S | 1193.64 | 0.03S | 721.29 | 0.03S |

Recordar que en el modelaje matemático esta hecho para ampl y que los vehículos no se toman en cuenta, ya que ampl hace ese calculo automáticamente.

Aun así en los códigos de las heurísticas estos estén dentro de las restricciones con la capacidad y la negación de vuelta a origen.

La heurística de Local Search toma en cuenta cada nodo, ya que a diferencia de las otras esta no toma en cuenta la distancia entre ellas, aunque puede ser un error de código que Greedy y NN dejen una solas en ciertas instancias.

De vez en cuando los algoritmos de Greedy y NN dejan afuera algunos nodos, dejando incompleta la entrega, ya que se excede la capacidad de carga de estas instancias

Mientras mas unidos los nodos, más común es que uno de estos quede afuera dentro de la heurística de greedy y NN

Mientras mas autos hay presentes, menos abarca la heurística Se logra apreciar en los E-n76-k

Los vehículos se deben decidir según la cantidad de nodos y la demanda de estos

En P-n55-k8 no se toma en cuenta el 8vo vehículo ya que con 7 recorre todos los clientes, pero una vez aplicando Local a NN, este logra un mejor resultado que P-n55-k7

En p-n76-k4 tiene una carga de 300 por vehículo y este deja nodos fuera de la ruta, mientras que p-n76-k5 tiene una carga de 280 y completa toda la ruta de nodos, aunque con un mayor costo.

El Local Search funciona mejor teniendo una función o heurística que ordene las lista al inicio

### Interacción y mejora

La implementación de las heurísticas en Python y la configuración de AMPL y LKH-3 se han concebido con la finalidad de representar fielmente el problema y explorar las capacidades de cada herramienta. La metodología experimental meticulosa, acompañada de métricas de evaluación definidas, facilitará un análisis de datos exhaustivo que se centrará en la comparación estadística de las soluciones y los tiempos de ejecución obtenidos por cada herramienta. Dichos análisis permitirán la identificación de diferencias significativas y la formulación de conclusiones precisas sobre las fortalezas y limitaciones inherentes a cada enfoque. Además, la adaptabilidad de la definición del problema en función de los resultados obtenidos ofrecerá un medio iterativo para perfeccionar futuros experimentos y refinar las implementaciones. Estas conclusiones, sustentadas por un análisis exhaustivo de datos, servirán como base para la propuesta de mejoras concretas en las estrategias de optimización de rutas, con el objetivo de optimizar la eficiencia y la efectividad.

## Definición del problema

El Problema de Ruteo de Vehículos Abierto (OVRP) presenta un desafío logístico significativo al requerir la planificación eficiente de rutas para una flota de vehículos, con el objetivo de satisfacer las demandas de un conjunto disperso de clientes geográficamente. A diferencia del Problema de Ruteo de Vehículos (VRP) convencional, en el OVRP, los vehículos no están obligados a retornar al depósito después de atender a los clientes, introduciendo así complejidades adicionales en el diseño de las rutas.

Este problema se clasifica como NP-Hard, indicando que a medida que el número de clientes o nodos en la ruta aumenta, el tiempo y el costo computacional para encontrar la ruta óptima crecen de manera exponencial, elevando la dificultad de encontrar soluciones precisas. Para abordar esta complejidad, se han empleado algoritmos heurísticos o de aproximación, como la destacada heurística Lin-Kernighan (LKH), reconocida por su efectividad en experimentos computacionales.

La importancia de estos algoritmos radica en su capacidad para generar soluciones buenas y factibles dentro de límites de tiempo de cómputo específicos (Lin, 1975). Su aplicación no solo ahorra costos logísticos, beneficiando a diversas empresas de transporte, sino que también ha impulsado investigaciones continuas en este campo. Entre los avances notables se encuentra LKH, así como herramientas de optimización poderosas como AMPL, que han demostrado su eficacia en la resolución de problemas logísticos complejos asociados al OVRP.

Es necesario comprender que, debido a las restricciones de eliminación de subtours (MTZ) en el modelo matemático, se busca identificar un ciclo hamiltoniano, lo que implica visitar nodos en un ciclo cerrado. Se establece una continuidad en las rutas, de modo que, si se entra en un nodo, es obligatorio salir de él. La introducción de la última restricción rompe esta continuidad, y aunque el modelo pueda ejecutarse, es probable que permanezca en ejecución durante un período prolongado sin encontrar una solución factible o, posiblemente, arroje resultados erróneos.

## Formulación del Modelo Matemático. (ARREGLAR)

El modelo matemático que representa el problema el problema de enrutamiento de vehículos abiertos se desarrolla mediante la modificación del modelo matemático para VRPB (Vehicle Routing Problem with Pickup and Delivery) propuesto por [3]. Esta adaptación implica realizar ajustes significativos en las restricciones del modelo para permitir mayor flexibilidad en las rutas de los vehículos. En primer lugar, se elimina la restricción de que cada ruta debe comenzar y terminar en el depósito, posibilitando que los vehículos inicien y finalicen en cualquier cliente. Además, se modifica la restricción relacionada con la carga y descarga para adaptarla al contexto de un OVRP, donde la carga total asociada con los clientes puede ajustarse para no exceder la capacidad del vehículo, sin la necesidad de regresar al depósito. La restricción de precedencia, que antes establecía un orden específico de visita para clientes de carga y descarga, se retira, permitiendo una mayor flexibilidad en las operaciones logísticas. Estos ajustes buscan optimizar la eficiencia de la planificación de rutas, minimizando la distancia total recorrida por los vehículos en consonancia con las demandas del problema adaptado.

**Conjuntos**

: Conjunto de nodos, incluye clientes y el depósito.

: Conjunto de arcos, representa la conexión entre el nodo y el nodo .

**Parámetros**

: Demanda de cada cliente.

: Capacidad del vehículo.

Matriz de costos, indica el costo de viajar de un no a otro.

**Variables**

: Variable binaria que es igual a 1 si el arco es utilizado y 0 de lo contrario.

: Variable continua que se utiliza en la eliminación de subtours

**Función Objetivo**

**Restricciones**

**Cada cliente es visitado una vez**

**Cada vehículo sale de un cliente una vez. A excepción del final.**

**Eliminación de subtours y restricciones de ruta final.**

**Restricción de capacidad**

# RESULTADOS Y DISCUSIÓN

En la discusión de los resultados obtenidos en la resolución del Problema de Ruteo de Vehículos Abierto (OVRP), se observa una variación en el rendimiento de los diferentes enfoques evaluados. La formulación en AMPL, aunque muestra resultados competitivos en comparación con el algoritmo LKH-3, evidencia diferencias más notables en algunas instancias. Este hallazgo sugiere que la formulación matemática propuesta es eficaz en la resolución del OVRP, aunque la elección del método de resolución puede depender de la naturaleza específica de cada instancia y de las restricciones de tiempo computacional.

Por otro lado, el algoritmo Greedy, utilizado de manera individual, presenta resultados notables, mientras que Nearest Neighbor (NN) junto con el método de optimización local 2-opt demuestran ser estrategias eficaces, proporcionando soluciones cercanas a las obtenidas por métodos más complejos como LKH-3 y AMPL. La capacidad de estas heurísticas para generar soluciones de buena calidad en tiempos computacionales más cortos sugiere su utilidad práctica, especialmente en escenarios logísticos donde la eficiencia en la planificación de rutas es esencial.

Es crucial destacar que la variación en la calidad de las soluciones entre instancias puede atribuirse a la naturaleza NP-Hard del OVRP, lo que implica que la complejidad del problema aumenta exponencialmente con el tamaño de la instancia. A pesar de las limitaciones inherentes a la búsqueda de soluciones precisas en problemas de esta índole, los resultados obtenidos subrayan la importancia de considerar enfoques heurísticos y técnicas de optimización avanzadas para abordar desafíos logísticos significativos. En conjunto, esta evaluación proporciona una perspectiva integral sobre la idoneidad de diferentes estrategias para la resolución del OVRP, considerando tanto la calidad de las soluciones como la eficiencia computacional en un contexto aplicado.

# CONCLUSION

# RECOMENDACIONES

En la búsqueda constante de mejorar la eficiencia y calidad en la planificación de rutas para el Problema de Ruteo de Vehículos Abierto (OVRP), se sugiere una serie de recomendaciones estratégicas. En primer lugar, se propone continuar la optimización del modelo matemático en AMPL, explorando ajustes detallados en restricciones y variables para adaptarse mejor a la complejidad inherente de los escenarios logísticos abordados. Además, se alienta a explorar enfoques híbridos que integren características de diversos algoritmos, capitalizando las fortalezas individuales de cada método y potenciando su rendimiento en conjunto.

Una recomendación clave es la realización de experimentos sistemáticos para ajustar los parámetros heurísticos de las estrategias Greedy, Nearest Neighbor, y 2-opt, permitiendo una optimización fina del rendimiento de estas heurísticas en casos específicos del OVRP. Asimismo, se sugiere la incorporación de métodos de búsqueda local avanzados, como simulated annealing o búsqueda tabú, para explorar soluciones de manera más efectiva en el espacio de búsqueda.

La evaluación en conjuntos de datos adicionales se presenta como una oportunidad valiosa para validar la robustez de las estrategias propuestas en diversos escenarios logísticos del mundo real. Un estudio de sensibilidad sobre los parámetros críticos del modelo y los algoritmos proporcionaría una comprensión más profunda de cómo pequeños ajustes impactan en la calidad de las soluciones y la eficiencia computacional.

Por último, se recomienda explorar la integración de tecnologías emergentes, como el aprendizaje automático o la optimización basada en inteligencia artificial, para abordar el OVRP. Estas tecnologías ofrecen enfoques innovadores y adaptativos que podrían proporcionar soluciones más flexibles y eficaces en entornos logísticos dinámicos. En conjunto, estas recomendaciones buscan impulsar la evolución continua de las estrategias utilizadas, asegurando soluciones cada vez más eficientes y adaptables para desafíos logísticos de alta complejidad.

# REFERENCES

1. Ren C., Research on Single and Mixed Fleet Strategy for Open Vehicle Routing Problem, Journal of Software, Vol. 6, Issue 10, 2011, pp.

2076–2081. https://doi.org/10.4304/jsw.6.10.20 76-2081.

1. Sedighpour M., Ahmadi V.,

Yousefikhoshbakht M., Didehvar F., and Rahmati F.,Solving the open vehicle routing problem by a hybrid ant colony optimization. vol. 41. 2014.

1. Toth, P., & Vigo, D. (Eds.). (Año de publicación). *Vehicle Routing: Problems, Methods, and Applications* (2da ed.). Society for Industrial and Applied Mathematics.

# Anexo de Codigos.

**Librerias Usadas**

import os

import re

import numpy as np

import time

import matplotlib.pyplot as plt

import random

**Greedy**

Procedimiento ext\_data(direccion):

Abrir archivo en modo lectura con dirección

nomb = ""

dim = 0

veh = 0

cap = 0

autos = []

x\_coord = []

y\_coord = []

dema = {}

coordenadas = Falso

demandas = Falso

Para cada línea en el archivo:

Si la línea comienza con "NAME":

nomb = extraer\_nombre(linea)

Si la línea comienza con "COMMENT":

veh = extraer\_numero\_vehiculos(linea)

Si la línea comienza con "VEHIC":

veh = extraer\_numero\_vehiculos(linea)

Si la línea comienza con "DIMEN":

dim = extraer\_dimension(linea)

Sino, si la línea comienza con "CAPAC":

cap = extraer\_capacidad(linea)

Sino, si la línea comienza con "NODE\_COORD\_SECTION":

coordenadas = Verdadero

Continuar al siguiente ciclo

Sino, si la línea comienza con "DEMAND\_SECTION":

coordenadas = Falso

demandas = Verdadero

Continuar al siguiente ciclo

Sino, si la línea comienza con "DEPOT\_SECTION":

demandas = Falso

Sino, si coordenadas es Verdadero:

coords = separar\_coordenadas(linea)

Si la longitud de coords es mayor o igual a 3:

Agregar coords[1] a x\_coord

Agregar coords[2] a y\_coord

Sino, si demandas es Verdadero:

demands = separar\_demandas(linea)

Si la longitud de demands es mayor o igual a 2:

nodo = convertir\_a\_entero(demands[0])

dem = convertir\_a\_entero(demands[1])

dema[nodo - 1] = dem

Para i en rango de veh:

Agregar (i+1) a autos

cap\_x\_veh = Crear diccionario con i desde 1 hasta veh con capacidad

Devolver nomb, dim, cap\_x\_veh, x\_coord, y\_coord, dema, autos

Procedimiento OVRP\_Greedy(vehiculos, clientes, distancia, q, Q):

rutas = Crear diccionario vacío

vehiculos\_disponibles = Copiar lista vehiculos

Mientras la longitud de clientes sea mayor que 0 y la longitud de vehiculos\_disponibles sea mayor que 0:

vehiculo = vehiculos\_disponibles[0]

capacidad\_actual = Q[vehiculo]

ruta\_actual = [0]

distancia\_recorrida = 0

Mientras capacidad\_actual sea mayor que 0 y la longitud de clientes sea mayor que 0:

cliente\_mas\_cercano = Encontrar cliente más cercano según distancia[(ruta\_actual[-1], cliente)]

Si q[cliente\_mas\_cercano] es menor o igual a capacidad\_actual:

capacidad\_actual -= q[cliente\_mas\_cercano]

distancia\_recorrida += distancia[(ruta\_actual[-1], cliente\_mas\_cercano)]

Agregar cliente\_mas\_cercano a ruta\_actual

Quitar cliente\_mas\_cercano de clientes

Sino:

Romper

rutas[vehiculo] = ruta\_actual

Quitar primer elemento de vehiculos\_disponibles

Devolver rutas

Procedimiento listar\_contenido\_directorio(ruta\_dic):

Intentar:

contenido = Obtener lista de archivos en ruta\_dic

Para cada elemento en contenido:

Obtener datos de archivo usando ext\_data

inst = nombre del archivo

inicio = Obtener tiempo actual

n = dim

clientes = Crear lista de clientes

nodos = Crear lista de nodos con 0 seguido de clientes

arcos = Crear lista de arcos con todas las combinaciones posibles de nodos distintos

q = dema

vehiculos = autos

Q = cap\_x\_veh

X = x\_coord

Y = y\_coord

distancia = Crear diccionario de distancias entre nodos

solucion = OVRP\_Greedy con parámetros vehiculos, clientes, distancia, q, Q

distancia\_total = 0

Para cada vehiculo, ruta en solucion:

distancia\_recorrida\_vehiculo = Sumar distancias entre nodos consecutivos en la ruta

Sumar distancia\_recorrida\_vehiculo a distancia\_total

fin = Obtener tiempo actual

tiempo\_transcurrido = fin - inicio

Imprimir información sobre la instancia, tiempo de ejecución y distancia total recorrida

Excepto Error de Permiso:

Imprimir mensaje de falta de permisos para acceder a ruta\_dic

ruta\_directorio = Obtener directorio de trabajo actual

Para cada raiz, directorios, archivos en os.walk(ruta\_directorio):

Para cada nombre\_directorio en directorios:

ruta\_completa = Unir raiz con nombre\_directorio

Llamar a listar\_contenido\_directorio con ruta\_completa

**Nearest Neighbor**

Abrir archivo con "INSTANCES/A/A-n45-k7.vrp"

nomb = ""

dim = 0

veh = 0

cap = 0

autos = []

x\_coord = []

y\_coord = []

dema = {}

coordenadas = Falso

demandas = Falso

Para cada línea en el archivo:

Si la línea comienza con "NAME":

nomb = Extraer nombre desde línea

Si la línea comienza con "COMMENT":

veh = Extraer número de camiones desde línea

Si la línea comienza con "VEHIC":

veh = Extraer número de camiones desde línea

Si la línea comienza con "DIMEN":

dim = Extraer dimensión desde línea

Sino, si la línea comienza con "CAPAC":

cap = Extraer capacidad desde línea

Sino, si la línea comienza con "NODE\_COORD\_SECTION":

coordenadas = Verdadero

Continuar al siguiente ciclo

Sino, si la línea comienza con "DEMAND\_SECTION":

coordenadas = Falso

demandas = Verdadero

Continuar al siguiente ciclo

Sino, si la línea comienza con "DEPOT\_SECTION":

demandas = Falso

Sino, si coordenadas es Verdadero:

coords = Separar coordenadas desde línea

Si la longitud de coords es mayor o igual a 3:

Agregar coords[1] a x\_coord

Agregar coords[2] a y\_coord

Sino, si demandas es Verdadero:

demands = Separar demandas desde línea

Si la longitud de demands es mayor o igual a 2:

nodo, dem = Convertir a entero demands[0] y demands[1] respectivamente

dema[nodo - 1] = dem

Para i en rango de veh:

Agregar (i+1) a autos

cap\_x\_veh = Crear diccionario con i desde 1 hasta veh con capacidad

n = dim

clientes = Crear lista de clientes desde 0 hasta n - 1

arcos = Crear lista de arcos con todas las combinaciones posibles de nodos distintos

vehiculos = veh

capacidad = cap

q = dema

x = x\_coord

y = y\_coord

distancia = Crear diccionario de distancias entre nodos usando la fórmula de la distancia euclidiana

partida\_tiempo = Obtener tiempo actual

Procedimiento Nearest\_Neighbor(nodo\_inicial, clientes, distancia, capacidad\_maxima=capacidad, nodos\_visitados=None):

Si nodos\_visitados es Nulo:

nodos\_visitados = Crear conjunto vacío

NN = [nodo\_inicial]

capacidad\_actual = 0

n = Obtener longitud de clientes

Agregar nodo\_inicial a nodos\_visitados

Mientras la longitud de NN sea menor que n:

k = Último elemento de NN

nn = Crear diccionario con pares (k, j) y distancias correspondientes para j en clientes no visitados

Si nn está vacío:

Romper

new = Obtener par mínimo desde nn según distancias

Si capacidad\_actual + q[new[0][1]] es menor o igual a capacidad\_maxima:

Agregar new[0][1] a NN

Incrementar capacidad\_actual por q[new[0][1]]

Agregar new[0][1] a nodos\_visitados

Sino:

Romper

Devolver NN

rutas = Crear lista vacía

nodos\_visitados\_global = Crear conjunto vacío

Para cada vehículo en rango de vehiculos:

NN = Llamar a Nearest\_Neighbor con nodo\_inicial 0, clientes, distancia y nodos\_visitados=nodos\_visitados\_global

Agregar NN a rutas

Unir nodos\_visitados\_global con NN

Para cada i, ruta en enumerar rutas:

Imprimir "Distancia de la Ruta {i + 1}: {calcular\_distancia\_ruta(ruta, distancia)}"

distancia\_total = Sumar todas las distancias en distancias\_rutas

final\_tiempo = Obtener tiempo actual

total\_tiempoNN = Restar final\_tiempo y partida\_tiempo

Imprimir "Suma Total de Distancias: {distancia\_total}"

Imprimir "El tiempo total de ejecucion de NN: {total\_tiempoNN}"

**Local Search**

Abrir archivo con "INSTANCES/F/F12.vrp"

nomb = ""

dim = 0

veh = 0

cap = 0

autos = []

x\_coord = []

y\_coord = []

dema = {}

coordenadas = Falso

demandas = Falso

Para cada línea en el archivo:

Si la línea comienza con "NAME":

nomb = Extraer nombre desde línea

Si la línea comienza con "COMMENT":

veh = Extraer número de camiones desde línea

Si la línea comienza con "VEHIC":

veh = Extraer número de camiones desde línea

Si la línea comienza con "DIMEN":

dim = Extraer dimensión desde línea

Sino, si la línea comienza con "CAPAC":

cap = Extraer capacidad desde línea

Sino, si la línea comienza con "NODE\_COORD\_SECTION":

coordenadas = Verdadero

Continuar al siguiente ciclo

Sino, si la línea comienza con "DEMAND\_SECTION":

coordenadas = Falso

demandas = Verdadero

Continuar al siguiente ciclo

Sino, si la línea comienza con "DEPOT\_SECTION":

demandas = Falso

Sino, si coordenadas es Verdadero:

coords = Separar coordenadas desde línea

Si la longitud de coords es mayor o igual a 3:

Agregar coords[1] a x\_coord

Agregar coords[2] a y\_coord

Sino, si demandas es Verdadero:

demands = Separar demandas desde línea

Si la longitud de demands es mayor o igual a 2:

nodo, dem = Convertir a entero demands[0] y demands[1] respectivamente

dema[nodo - 1] = dem

Para i en rango de veh:

Agregar (i+1) a autos

cap\_x\_veh = Crear diccionario con i desde 1 hasta veh con capacidad

n = dim

clientes = Crear lista de clientes desde 0 hasta n - 1

arcos = Crear lista de arcos con todas las combinaciones posibles de nodos distintos

vehiculos = veh

capacidad = cap

q = dema

x = x\_coord

y = y\_coord

distancia = Crear diccionario de distancias entre nodos usando la fórmula de la distancia euclidiana

Procedimiento asignar\_sublistas(clientes, demanda, vehiculos, capacidad):

Ordenar clientes por demanda en orden descendente

Inicializar listas para asignar clientes a cada vehículo

capacidad\_actual = [0] \* vehiculos

Para cada cliente en clientes\_ordenados:

Encontrar vehículo con capacidad disponible

Verificar si la asignación cumple con la capacidad del vehículo

Si capacidad\_actual[vehiculo\_disponible] + demanda[cliente] <= capacidad:

Agregar cliente a asignaciones[vehiculo\_disponible]

Incrementar capacidad\_actual[vehiculo\_disponible] por demanda[cliente]

Sino:

Encontrar otro vehículo con capacidad disponible

Agregar cliente a asignaciones[vehiculo\_disponible]

Incrementar capacidad\_actual[vehiculo\_disponible] por demanda[cliente]

Eliminar el 0 de cada lista y colocar un 0 al inicio de cada lista

Devolver asignaciones

rutas = Llamar a asignar\_sublistas con clientes, q, vehiculos y capacidad

Procedimiento calcular\_distancia\_ruta(ruta, distancia):

distancia\_total = 0

Para i en rango de longitud de ruta - 1:

Incrementar distancia\_total por distancia[(ruta[i], ruta[i + 1])]

Devolver distancia\_total

distancias\_rutas = Crear lista de distancias para cada ruta en rutas

Para cada i, distancia\_ruta en enumerar distancias\_rutas:

Imprimir "Distancia de la Ruta {i + 1}: {distancia\_ruta}"

distancia\_total = Sumar todas las distancias en distancias\_rutas

Imprimir "Suma Total de Distancias: {distancia\_total}"

partida\_tiempo = Obtener tiempo actual

Procedimiento Local\_Search(NN, distancia):

min\_cambio = 0

Para cada i en rango de longitud de NN - 2:

Para cada j en rango de i + 2 hasta longitud de NN - 1:

costo\_actual = distancia[(NN[i], NN[i+1])] + distancia[(NN[j], NN[j+1])]

costo\_nuevo = distancia[(NN[i], NN[j])] + distancia[(NN[i+1], NN[j+1])]

cambio = costo\_nuevo - costo\_actual

Si cambio < min\_cambio:

min\_cambio = cambio

min\_i = i

min\_j = j

Si min\_cambio < 0:

Invertir elementos de NN desde min\_i+1 hasta min\_j+1

Devolver NN

contador = 0

Mientras Verdadero:

contador += 1

primera = Sumar distancias de cada ruta en rutas

Para cada i en rango de longitud de rutas:

ruta\_anterior = Copiar rutas[i]

rutas[i] = Llamar a Local\_Search con rutas[i] y distancia

dist\_actual = Calcular distancia de ruta\_anterior

dist\_nueva = Calcular distancia de rutas[i]

ultima = Sumar distancias de cada ruta en rutas

intercambio = Obtener valor absoluto de (ultima - primera)

Si intercambio == 0:

Romper el bucle

time\_f = Obtener tiempo actual

fin\_timeLocal = Restar time\_f y partida\_tiempo

Imprimir "Solucion", rutas

Imprimir "Distancia Total", Sumar distancias de cada ruta en rutas

Imprimir "Total de Intentos", contador

Imprimir "Tiempo de ejecucion de Local Search:", fin\_timeLocal