ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ciencias Naturales y Matemáticas

Una metaheurística para resolver el problema de ruteo vehicular capacitado en empresas de distribución secundaria

PROYECTO INTEGRADOR

Previo a la obtención del Título de:

Matemático

Presentado por:

JOSE DANIEL TELLO PALMA

GUAYAQUIL - ECUADOR

Año: 2022

DEDICATORIA

Dedico este trabajo a mis padres, Lorenzo y Rosario por su sacrificio y esfuerzo para lograr esta meta.

JOSE TELLO P.

AGRADECIMIENTOS

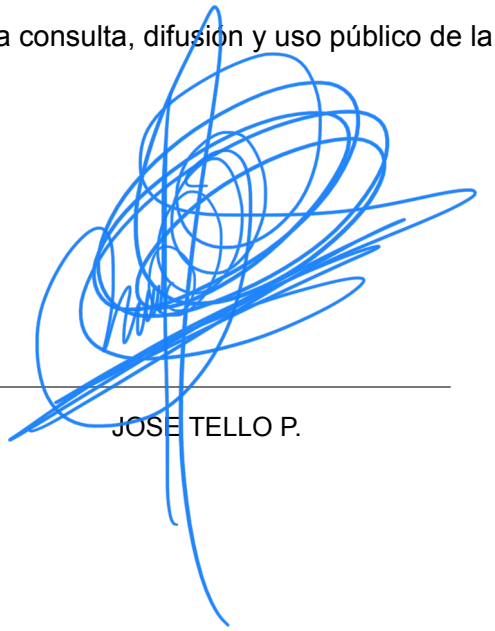
Profundos agradecimientos al claustro de profesores e investigadores extranjeros de la Facultad de Ciencias Naturales y Matemáticas, que en su inicio se dio lugar a inaugurar la carrera de Matemáticas siendo un personal competente y calificado, y consecuentemente dando la oportunidad a jóvenes soñadores a formarse. A PhD. Domingo Quiroz, por sus consejos, confianza y sabiduría; a Ph.D. Danny de Cecchis y Ph.D Cristhian Galarza por su acentuado afán de enseñanza; a Ph.D Elimar Marchan, por su labor administrativa coordinando el grado.

A mi amigo, Kevin Chica, mis compañeros Alexander P. y Franklin B, por su soporte.

JOSE TELLO P.

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; *JOSE DANIEL TELLO PALMA,* y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”



JOSE TELLO P.

EVALUADORES

Firmado electrónicamente por:
LUZ ELIMAR
MARCHAN
MENDOZA

Ph.D. Luz Marchan Mendoza

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
DAVID ANTONIO
DE SANTIS
BERMEO

Msc. David Antonio De Santis

PROFESOR TUTOR

RESUMEN

El problema de ruteo vehicular capacitado (más conocido por sus siglas en inglés CVRP) es uno de los problemas elementales en la gestión de la cadena de suministro. CVRP está entre los problemas NP-difícil, y muchos algoritmos han sido desarrollados para resolver instancias grandes en un tiempo aceptable, su complejidad es también debido a que contiene tanto el problema del embalaje y el problema del agente viajero (más conocido como TSP por sus siglas en inglés) como casos especiales, siendo también el TSP un problema NP-difícil y uno de los problemas de la optimización combinatoria más conocidos en los campos de logística, transporte, y distribución. La metaheurística de recocido simulado apoyado por enfoques heurísticos de búsqueda local es llevado a cabo para resolver estos problemas. Los resultados computacionales para algunos datos de referencia son reportados resaltando la calidad de las soluciones obtenidas y el tiempo de ejecución. Un breve contraste del desempeño de los lenguajes de programación de Python y Matlab es presentado para el TSP. Este trabajo además presenta un estudio sistemático y cronológico del trasfondo teórico junto a sus resultados recientes por la comunidad científica.

Palabras Clave: VRP, TSP, Recocido Simulado, NP-difícil, 2-opt.

ABSTRACT

The capacitated vehicle routing problem (better known by its acronym CVRP) in on of the elemental problem in supply chain management. CVRP is among the NP-hard problems, and many algorithms have been developed to solve large instances in an acceptable time, its complexity is also due to it contains both the bin packing problem and the travelling salesman problem (better known by its acronym TSP) as special cases, being TSP a NP-hard problem as well and one of the well-known combinatorial optimization problems in the fields of logistics, transportation, and distribution. A simulated annealing metaheuristic supported by local search heuristic approach is carried out to solve kind of problems. Computational results of some benchmarks are reported highlighting quality of solutions and elapsed time. A brief contrast of the performance of the Python and MATLAB programing languages is shown for TSP. This work also present a systematic and chronological study is presented for the theoretical background and current results by the scientific community.

Keywords: *VRP, TSP, Simulated Annealing, NP-hard, 2-opt.*

ÍNDICE GENERAL

ABREVIATURAS

CVRP	Capacited VRP
	Capacitado VRP
DCVRP	Distance-constrained Capacited VRP
	VRP capacitado con restricción de distancia
ESPOL	Escuela Superior Politécnica del Litoral
GRASP	Greedy randomized adaptive search procedure
	Procedimientos de busqueda miopes aleatorizados y adaptativos
NN	Nearest Neighbor
	Vecino más cercano
PVRP	Periodic VRP
	VRP periódico
SA	Simulated Annealing
RC	Recocido Simulado
SDVRP	Split Delivery VRP
	VRP de entrega dividida
SIAM	Society for Industrial and Applied Mathematics
	Sociedad de Matemáticas Aplicadas e Industriales
TSP	Travelling Salesman Problem
	Problema del agente viajero
VRP	Vehicle Routing Problem
	Problema de ruteo vehicular
VRPM	VRP with Multiple use of vehicles
	VRP con uso múltiple de vehículos
VRPPD	VRP with Picksups and Deliveries
	VRP con recogidas y entregas

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

La integración de la industria logística, las tecnologías de la información, la tendencia de la globalización y toda la distribución logística ocupa un rol más relevante con el pasar de los años. De los sistemas de distribución, el subsistema más importante es el de transporte. Optimizar su logística y reducirle el costo es una de las formas efectivas para que las empresas, especialmente las de distribución logística, mejoren su competitividad.

Más de 60 años han transcurrido desde que Dantzig and Ramser (1959) introdujeron el VRP (Problema de ruteo de vehículos), en aquel momento conocido como problema del despacho de camiones, como una aplicación del mundo real sobre la entrega de gasolina a estaciones de servicio. De esta forma, los autores propusieron la primera formulación de programación matemática y un enfoque algorítmico para el VRP. No transcurrió mucho tiempo, hasta que 3 años más tarde un primer borrado fuera presentado para que finalmente Clarke and Wright (1964) introdujeran el concepto de ahorros, proponiendo una efectiva y codiciosa heurística para resolver el VRP, y así finalmente mejorar la formulación del problema que sus antecesores ya habían iniciado.

Siendo uno de los desafíos más que hacen frente las compañías de logística, la comunidad científica rápidamente captó la atención del problema y, apoyándose en los brillantes precursores, se han publicado un sin número de artículos durante las últimas 6 décadas en las revistas internacionales *Operations Research* y *Trannsportation Science* presentando modelos matemáticos y proponiendo métodos exactos como algoritmos de

diseño, programación dinámica; métodos meta-heurísticos, entre los que cabe mencionar búsqueda tabú, recocido simulado, algoritmos genéticos, algoritmo de hormigas, búsqueda en vecindades variables, redes neuronales, entre otros. Este último tipo es una alternativa a modelos de alto coste computacional que se mantiene adaptándose para hacer frente a una variedad de limitaciones. Adicionalmente, en las revistas que regularmente publican artículos sobre VRP se hace mención a *Computers Operations Research*, *EURO Journal on Transportation and Logistics*, *Journal of the Operational Research Society*, *Transportation Research*, *Networks*, y *Journal of Heuristic*. La resolución computacional de un VRP y su clasificación es NP-difícil según Lenstra and Kan (1981), porque los problemas del mundo real involucran restricciones complejas como limitaciones de carga, ventanas de tiempo, tiempos de viaje dependientes del tiempo (que reflejan la congestión del tráfico), múltiples depósitos y flotas heterogéneas. Estas características introducen una complejidad significativa y han hecho evolucionar drásticamente el panorama de investigación de VRP.

En este trabajo, se pretende considerar una aproximación de la solución mediante el recocido simulado al problema de ruteo de vehicular capacitado, que bien podría o no hallar una solución óptima. Esto es, una solución orientada a optimizar los tiempos de entregas y/o distribución sobre una red rutas de vehículos.

1.1 Justificación del problema

Asegurar el flujo continuo de productos desde el productor hasta el consumidor final es una tarea fundamental en la gestión de las cadenas de suministro. Cuando los problemas de tipo VRP se abordan de manera efectiva, la gestión del transporte puede hacerse más sostenible, lo que da como resultado un sistema de distribución más eficiente. Por ello, se diseñan diversos modelos matemáticos y algoritmos en la búsqueda de soluciones no sólo para reducir los

costos de transporte, sino también para encontrar el menor costo posible basado en el uso eficiente de los recurso del área de distribución. El objetivo para las compañías es mantener los costos bajos, implícitamente aumentar las ganancias, mientras conservan la calidad del producto.

En la actualidad, el transporte suele ser un componente significante del costo de un producto alcanzando un 10%, por lo que minimizar los costos de distribución puede suponer un ahorro de hasta el 5% para una empresa mediante el uso de programas informáticos de optimización, según Paolo Toth (2014). Por otro lado, es importante resaltar el peso que tiene la industria del transporte sobre el Producto Interno Bruto, pues bien, para la Unión Europea el sector del transporte representa hasta un 10% del PBI; de modo que según Hasle et al. (2007), un ahorro incluso menos del 5% representaría cuantiosas utilidades a ser administradas. De esta forma, considerando que los mayores avances en materia de ciencia y tecnología se encuentra sobre los países más desarrollados, y las grandes compañías que a su vez se reservan el derecho de sus avances técnicos con el fin de generar alguna rentabilidad.

Por otro lado, la situación en Ecuador es diferente, si bien es escasa la investigación debido a los pocos fondos destinados a ésta, el país toma un rol de no generar sino de comprar los software y solvers. El presente proyecto busca una solución efectiva para el problema del ruteo vehicular que reduzca los costos de transporte. A su vez, esto ayuda a empresas a tener un crecimiento sostenible, mejora la productividad, eficiencia de la flota, y tratando de cortar la brecha con otros países desde el sector privado.

1.2 Objetivos

1.2.1 Objetivo General

Evaluar la metaheurística de recocido simulado en las instancias de Augerat mediante un lenguaje de programación que permita resolver el problema de ruteo vehicular capacitado.

1.2.2 Objetivos Específicos

A continuación se detallan los diferentes objetivos específicos del trabajo.

- Implementar la metaheurística de recocido simulado empleando el sistema de cómputo numérico Matlab;
- Evaluar la eficiencia de la metaheurística comparando las soluciones obtenidas con otras soluciones expuestas en revistas de carácter científico;
- Resolver el problema del agente viajero mediante el recocido simulado.

1.3 Marco teórico

1.3.1 Antecedentes

En 1959, el matemático George Dantzig junto a John Ramser, publican el artículo titulado ”The Truck dispatching problem” o bien ”El problema de despacho de camiones” en español, donde se le define por primera vez al VRP y tal cómo se especifica puede ser considerado como una generalización al Problema del Agente Viajero. Aquí se plantea una aproximación algorítmica para un problema en particular de entregas a estaciones de gasolina, donde el contexto es la entrega de bienes ubicados en un depósito inicial a clientes que han realizado una demanda de dichos productos con el objetivo de minimizar los costos de transporte. Por su parte, G. Clarke y J. Wright presentan un primer borrador en 1962, que más tarde lograrían publicar y

fue una mejora al enfoque de Dantzig propuesto años atrás, pues según (Clarke and Wright, 1964, p. 569), ”éste método tendía a poner más énfasis en llenar los camiones hasta casi su capacidad total que en minimizar las distancias”.

Durante 20 años, la comunidad científica intentó resolver este modelo matemático mediante el uso de heurísticas, puesto que métodos exactos aún no habían sido formalmente propuestos. Es así que, debido a Christofides et al. (1981a,b) se propone un algoritmo basado en programación dinámica con relajación del espacio de estado, análogo a la relajación Lagrangiana en programación entera, y además proponen dos formulaciones usando árboles de expación de k-mínimo y mínima q-caminos. Por otro lado, Laporte and Nobert (1987) notan que el uso de métodos exactos es viable en dimensiones relativamente modestas, esto debido a las limitadas capacidades de cómputo de aquella época. Dada el aumento de métodos y enfoques propuesto por la comunidad científica, Laporte (1992a,b) muestra una visión general de los algoritmos exactos y heurísticos tanto para el TSP y VRP desarrollados durante esa época. En la actualidad, Mor and Speranza (2022) exhiben los principales esfuerzos durante más de 60 años de trabajo conjunto de los investigadores.

1.3.2 Estado del Arte

1. Sanabria Quiñonez (2018) publica *Estudio comparativo de algoritmos basados en metaheurísticas aplicados a la solución del problema de ruteo de vehículos con capacidad limitada*, como tesis de maestría para ESPOL. Se pretendió hacer una comparativa de los algoritmos heurísticos de Clarke- Wright y barrido, junto a las metaheurísticas GRASP y RC, para ser aplicados a un caso de estudio en la empresa COTAIN S.A., quién tiene clientes a lo largo de la ciudad de Guayaquil, Daule y Samborondón del cantón Guayas, y su planta ubicada en el extremo norte de Pascuales en Guayaquil. Para este fin, la empresa sólo cuenta con dos vehículos. Sin

embargo, logró una disminución de 15 kilómetros respecto a la metodología usada por la organización.

2. Barcia Kleber (2019) asegura haber propuesto un modelo matemático para una empresa farmacéutica para la ciudad de Guayaquil-Ecuador, con el objetivo de optimizar y reducir los costos logísticos. Para esto, se emplea una variante del problema de ruteo vehicular con ventanas de tiempo, es decir, los bienes de entrega tienen un horario establecido a ser facilitados. Aquí presentan el uso de una heurística para mejorar la calidad del servicio en los hospitales públicos y privados, y clínicas. En consecuente, mediante el algoritmo Clark-Wright se evidenció una reducción del 45% de los recursos.

1.3.3 Formulación Matemática: CVRP

De acuerdo a Redi et al. (2020), la formulación matemática del Capacited-VRP es descrita por el siguiente modelo:

Inputs- Entradas:

- N: Conjunto de nodos donde $\{1, ..., n\}$ son clientes, y 0 es un depósito inicial;
- c_{ij}: Costo asociado[1], desde el nodo i al nodo j con $i, j \in N$
- κ: Número de vehículos disponibles.
- Q: Capacidad del vehículo
- d_i: Demanda del cliente i, con $i \in N$

Variables de decisión:

[1] El costo asociado puede verse reflejado por el tiempo de trasladarse desde $i \leftrightarrow j$, o bien por la distancia entre éstos.

- $x_{ij} = \begin{cases} 1 & \text{si el tramo } i \leftrightarrow j \text{ forma parte de una ruta óptima} \\ 0 & \text{caso contrario} \end{cases}$
- u_i, u_j: Variables utilizadas para aplicar la eliminación de la subruta.

Función Objetivo:

$$min \quad \psi = \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{1.1}$$

Sujeto a

$$\sum_{i \in N; i \neq j} x_{ij} = 1, \quad \forall j \in N \backslash \{0\} \tag{1.2}$$

$$\sum_{j \in N; i \neq j} x_{ij} = 1 \quad \forall i \in N \backslash \{0\} \tag{1.3}$$

$$\sum_{j \in N \backslash \{0\}} x_{0j} \leqslant \kappa \tag{1.4}$$

$$u_j - u_i + Q * x_{ij} \leqslant Q - d_i \quad i \neq j; \forall i, j, \in N \backslash \{0\} \tag{1.5}$$

$$d_i \leqslant u_i \leqslant Q \quad \forall i \in N \backslash \{0\} \tag{1.6}$$

$$u_i \geqslant 0, u_j \geqslant 0 \quad \forall i, j \in N \tag{1.7}$$

La función objetivo establecida en 1.1, describe la distancia total de recorrida por los vehículos durante una ruta. La variable de decisión x_{ij}, es 1 cuando un vehículo determinado se mueve desde el nodo i hacia el nodo j, 0 caso contrario; es usada en 1.2, 1.3 parar asegurar que cada desde un $i - simo$ nodo nos dirigamos hacia un único nodo j, y que desde un nodo j sólo puede llegarse desde un único nodo i. Dado que cada vehículo será retornado a su punto de partida luego de hacer las respectivas paradas durante su ruta, 1.4 nos asegura que el número de rutas no exceda el número de camiones disponibles. Además, y establece que la capacidad

de carga de cada vehículo no sea excedida. Finalmente, la restricción 1.7 es la eliminación de subrutas que determina que la ruta lograda sea un camino hamiltoniano.

1.3.4 TSP: Travelling Salesman Problem

El problema del agente viajero, más conocido como ”Travelling Salesman Problem” o TSP por sus siglas en inglés, es un caso particular del modelo matemático empleado en el CVRP. Aquí, el modelo se compone haciendo uso de 1.1, 1.2, 1.3, teniendo un único vehículo que realizará una única ruta. Detrás de las ecuaciones, su planteamiento es muy sencillo. Dado un conjunto de nodos o *ciudades*, *estaciones, o clientes* de tamaño *n*, y si *0* es una ciudad a elección, encontrar el camino de menor distancia que empieza en esta ciudad inicial y atraviesa cada una de las demás ciudades. Es decir, encontrar una ruta que pase a través de todos los nodos, una única vez, con la menor distancia posible.

TSP es un representante de una gran clase de problemas conocidos en optimización combinatoria, y en los campos de la logística, distribución y transporte. Entre ellos, el TSP es uno de los más importantes, ya que es muy fácil de describir, pero *muy difícil* de resolver. Por *muy difícil* hacemos referencia a su complejidad computacional, ya que pertenece a la clase *NP-Hard* (*NP-difícil*).

Los esfuerzos de la comunidad científica en intentar resolver instancias del problema del agente viajero se ve reflejado en los mayores logros expuesto a la fecha por Applegate et al. (1995, 2011) , mediante el desarrollo de un software, de nombre Concorde, basado en la programación entera para encontrar las rutas óptimas. Resolvió, entre muchos, un problema del vendedor ambulante que modela la producción de placas de circuito impreso que tienen 7 397 orificios (ciudades), un problema sobre las 13509 ciudades más grandes de los EE. UU., uno sobre las 24 978 ciudades de Suecia y, finalmente, un problema de 85 900 ciudades que surge de una aplicación VLSI.

1.4 Variantes de VRP

Dada la complejidad de problemáticas industriales y logísticas, se ha forzado la aparición de variantes del VRP. Esta las podemos clasificar de acuerdo a Paolo Toth (2014):

- **Entrega y Recolección, VRPPD:** Cuando un número determinado de bienes deben ser recogidos desde una ubicación para ser entregados en otras estaciones.
- **Entrega Dividida, SDVRP:** Una demanda de un cliente pueda ser dividida en demandas más pequeñas arbitrarias que sean atendidas por diferentes vehículos.
- **Suministro Periódico, PVRP:** Algunos bienes deben ser entregados más de una vez a un localización en particular.

Cuando las restricciones son internas a la ruta:

- **Capacidad de carga, CVRP:** Los vehículos tienen una límite de carga.
- **Longitud de ruta, DCVRP:** Existe un límite de distancia posible a ser recorrido por un vehículo.
- **Uso múltiple de vehículos, VRPM:** Algunos de los vehículos puede realizar más de una ruta
- **Ventanas de Tiempo, VRPTW:** Los productos deben ser entregados en un determinado horario de tiempo de acuerdo a las expectativas de los clientes.

Algunas variantes pueden ser clasificadas acorde a cualidades de flota:

- **Múltiple Depósito, MDVRP:** El punto de partida y final de ruta para algunos vehículos no son el mismo.

- **Flota mixta o heterogénea, HFVRP:** Este modelo surge cuando la flota de vehículos difiere en capacidad de carga, velocidad, y costos en el transporte.

De igual forma, cabe recalcar que el ruteo puede ser del tipo:

- **Dinámico:** La información relevante sobre las condiciones del sistema están disponibles durante el funcionamiento.
- **Estocástico:** Si las condiciones del sistema son inciertas, pero la ésta incertidumbre es descrita mediante una distribución de probabilidad.

1.5 Complejidad Computacional

El VRP es *NP-duro* en el sentido *fuerte*, ya que contiene al TSP y al problema de Bin Packing (BPP) como casos especiale, que también son NP-duro. Por NP-duro, o NP-difícil, entendemos que el problema pertenece a la clase dureza NP, dureza en el tiempo polinomial no determinista, de la teoría de las complejidades computacionales. Un algoritmo eficiente para el VRP, o para el TSP, es decir un algoritmo de computación, que por cualquier instancia de *n* nodos, tendremos la ruta de menor costo posible en tiempo polinomial. Esto, probablemente no existe. Más precisamente, tal algoritmo existe si y sólo si las dos clases computacionales P y NP coinciden, algo poco probable a lo desarrollado en investigación en los últimos años.

De acuerdo a Lenstra and Kan (1981), un problema E es NP-difícil, si hay un problema NP-completo F, tal que F es reducible a X en tiempo polinomial. Sin embargo, cualquier problema NP-completo puede reducirse a cualquier otro problema NP-completo en tiempo polinomial. De haber una solución para un problema NP-duro en tiempo polinomial, existe una solución para todos los problemas NP en tiempo polinomial, lo que daría por sentado el problema del milenio *P vs NP*, el cual tienen un premio de $\$1.000.000$ dólares

estadounidenses a la persona o grupo que demuestre si $P = NP$ o bien si $P \neq NP$ La mayoría de los matemáticos piensan que $P \neq NP$. El problema fue inicialmente introducido por Cook (1971), y más tarde ser presentado por el Instituto Clay de Matemáticas como ”El problema P vs NP” en 2000. Algunas de las implicaciones del problema, tendría grandes consecuencias positivas en investigación operativa, logística, criptografía, biología y filosofía descritos por Massacci and Marraro (2000); De et al. (2007); Horie and Watanabe (1997); Berger and Leighton (1998); Aaronson (2013).

> A medida que reflexionamos sobre P frente a NP, vemos que la pregunta tiene muchos significados diferentes. Está P vs. NP, la cuestión matemática: formalmente definida, obstinadamente abierta y aún con una recompensa de un millón de dólares por su cabeza. Hubo momentos en los que pudimos ver un camino a seguir para resolver P vs. NP a través de herramientas de teoría de la computabilidad, circuitos, pruebas y geometría algebraica. Por el momento, no tenemos un camino sólido para resolver el problema P vs NP. En cierto sentido, estamos más lejos de resolverlo que nunca. (Fortnow, 2021, p. 84.)

1.5.1 Desafíos Actales: Reto de CVRP

Con el fin de motivar aún más la investigación sobre métodos exactos para el CVRP, se ofrecen dos premios monetarios y el reconocimiento por resolver instancias del nuevo benchmark propuesto por Uchoa et al. (2017).

- Premio de los 500 clientes: resolver de manera óptima las 68 instancias de la serie X con hasta 500 clientes.
- Premio de los 1000 clientes: resolver de manera óptima las 100 instancias de la serie X

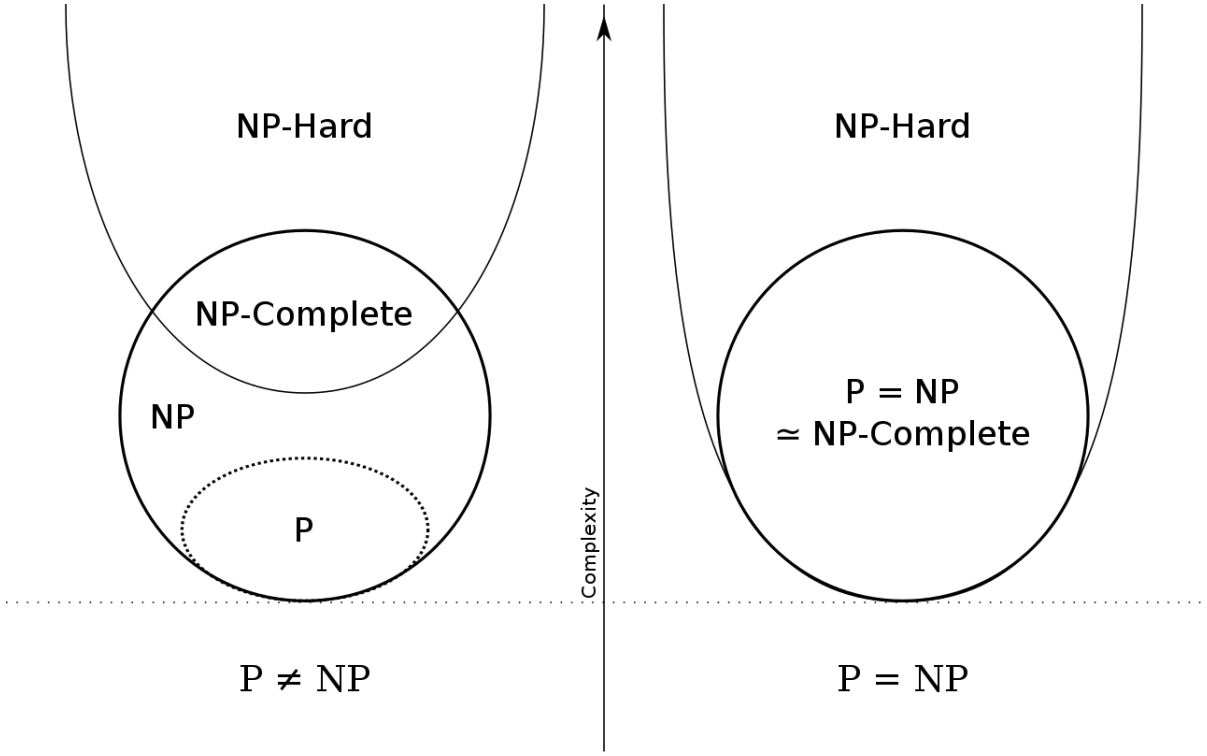


Figura 1.1. Diagrama de Euler para el conjunto de problemas P, NP, NP-completo y NP-duro (excluyendo el lenguaje vacío y su complemento, que pertenecen a P pero no son NP-completos)

con hasta 1000 clientes.

El premio está apoyado por la colaboración de la Universidad Pontificia Católica de Río de Janeiro, Universidad Federal Fluminense, Universidad Federal de Paraíba, Para esto, para reclamar el premio se debe tener en una las siguientes consideraciones:

1. El premio sólo se otorgará después de que el artículo correspondiente sea revisado por pares y aceptado para su publicación en una importante revista de investigación operativa.

2. El método debería ser capaz de resolver todas las instancias requeridas (68 para los premios de 500 clientes y 100 para los premios de 1000 clientes). No basta con resolver las últimas instancias nunca resueltas por otros autores.

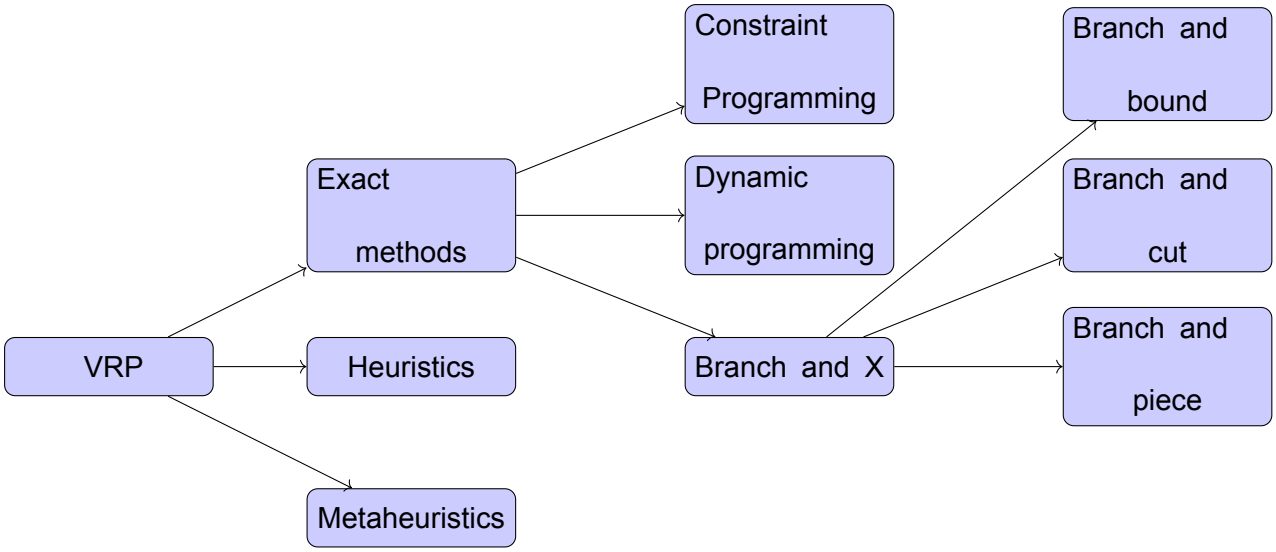


Figura 1.2. Métodos de Resolución para el VRP

3. No hay restricción del uso de computación en paralelo de CPUs.

1.5.2 Métodos de Resolución

Dada la importancia del problema en la industria, métodos para buscar la solución de un VRP dado se han desarrollado con el pasar de los años. De esta forma, la Sociedad de Matemática Industrial y Aplicada, SIAM por sus siglas en inglés, publica vía Paolo Toth (2014) una diversa lista de métodos con diferentes enfoques. Por otro lado, Laporte and Nobert (1987) y Tan and Yeh (2021) clasifican estos métodos, que podemos resumirlos en el diagrama de 1.2

Los métodos exactos, también llamados métodos formales, proporcionan soluciones óptimas. No obstante, encontrar la solución en un tiempo razonable de cómputo no está garantizado, debido a que VRP y sus variantes, son problemas del tipo NP-hard (incluso el TSP), e intentar resolver instantes más grandes de forma óptima va a requerir de mayor tiempo de cómputo propuesto. Por ende, solo suelen ser adecuados a pequeña escala, esto es, no superior a 200 nodos.

Por otro lado, heurísticas y las metaheurísticas brindan en general resultados casi

óptimos. Por el contrario, no hay límites en cuanto al tamaño del problema y aún así se puede manejar de forma eficiente una gran cantidad de restricciones y todavía generar soluciones casi óptimas.

Existen tres técnicas metaheurísticas ampliamente usadas para la optimización del VRP, estas son: Tabu Search (Búsqueda Tabú), Genetic Algorithm, Simulated Annealing (Recocido Simulado).

La Búsqueda Tabú es un método que guía a una heurística de búsqueda local a explorar el conjunto de soluciones más allá de su optimización local. Su bases son propuestas por primera vez en Glover (1986), que más tarde sería concretado en dos publicaciones en 1989 y 1990, estableciendo ideas fundamentales para superar las limitaciones de optimalidad tanto en frameworks deterministas y probabilísticos, y nuevas formas de gestión de las listas tabús.

Por otro lado, la metaheurística basada en un algoritmo genético se centra en explorar varias partes de la región factible y evoluciona gradualmente hacia mejores soluciones factibles. Su fundamento se inspira en el proceso de evolución biológica y su base genético molecular. Esta técnica ha sido en un caso de estudio famoso para Intel Corporation, según Rash and Kempf (2012), donde el corazón algorítmico de un sistema de soporte de decisión es un algoritmo genético enfocado en administrar las limitaciones de recursos, planificación y la optimización financiera. Este algoritmo genético permitió un nuevo proceso comercial para Intel Corporation, la empresa líder mundial en semiconductores. A su vez, en Tasan and Gen (2012) vemos una aplicación en concreto al problema de ruteo vehicular.

1.5.3 Recocido Simulado

El algoritmo de Recocido Simulado (también conocido como enfriamiento simulado) es una metaheurística que realiza una búsqueda local aleatoria para llegar a una solución casi óptima en problemas de optimización continua y combinatoria. No obstante, su fundamento yace en

Estado físico	Problema de Optimización
Estado del sistema	Solución
Posiciones moleculares	Variables de decisión
Energía	Función Objetivo
Estado fundamental	Solución óptima global
Estado meta-estable	Óptimo local
Enfriamiento rápido	Búsqueda local
Temperatura	Parámetro de control T
Recocido cuidadoso	Recocido Simulado

Tabla 1.1. Analogía del sistema físico y un problema de optimización del algoritmo meta-heurístico de RC. Fuente: Talbi (2009)

la mecánica estadística y el comportamiento de los sistemas físicos en la presencia de calor. Es una técnica que involucra el calentamiento y enfriamiento controlado de un material para alterar sus propiedades físicas. Este enfoque fue aplicado por primera vez para encontrar una solución al problema del agente viajero por Kirkpatrick et al. (1983), quiénes le adjuntaron el nombre de recocido simulado.

Para esto, la adaptación del algoritmo, cuyo origen se da en la metalurgia, hacia un problema de optimización la podemos observar en la tabla 1.1 establecida por Talbi (2009). RC es un algoritmo estocástico que permite la degradación de una solución bajo condiciones específicas. El objetivo es escapar del óptimo local y así retrasar la convergencia. Además, el algoritmo no almacena información recopilada durante la búsqueda local.

El algoritmo parte de una solución inicial x_1 y se mueve en cada iteración t desde x_t hacia una nueva solución de ruta x_{t+1} en la vecindad $N(x_t)$ de x_t, y se finaliza el proceso hasta que una condición sea satisfecha. Si $f(x)$ denota el costo total de la ruta x, entonces el costo

de la nueva ruta $f(x_{t+1})$ no será necesariamente menor que $f(x_t)$.

Por consiguiente, de acuerdo a Paolo Toth (2014), en cada iteración t del recocido, una solución aleatoria x es establecida en $N(x_t)$ y si $f(x) \leqslant f(x_t)$, entonces x_{t+1} pasa a ser x, de otra forma se tiene:

$$x_{t+1} := \begin{cases} x & \text{with probability } p_t \\ x_t & \text{with probability } 1 - p_t \end{cases} \tag{1.8}$$

de dónde, p_t es una función decreciente de t y de $f(x) - f(x_t)$. En particular, la distribución de Boltzmann, en honor a Boltzmann (1868), dada por:

$$p_t = e^{(-[f(x)-f(x_t)]/\theta_t)} \tag{1.9}$$

donde θ_t denota la *temperatura* en la iteración t. La regla empleada para definir a θ_t es llamada la regla de enfriamiento, descrita por:

$$\theta_{t+1} = \alpha \cdot \theta_t \tag{1.10}$$

De aquí que, la probabilidad de aceptar una solución equivocada debería decrecer con el tiempo. Tenemos 3 diferentes criterios: El valor de f no ha decrecido por al menos un número consecutivo k_1 de ciclos de T iteraciones; el número de movimientos aceptados ha sido menor que k_2 ciclos; y, $k3$ ciclos de T iteraciones (tope) han sido ejecutados.

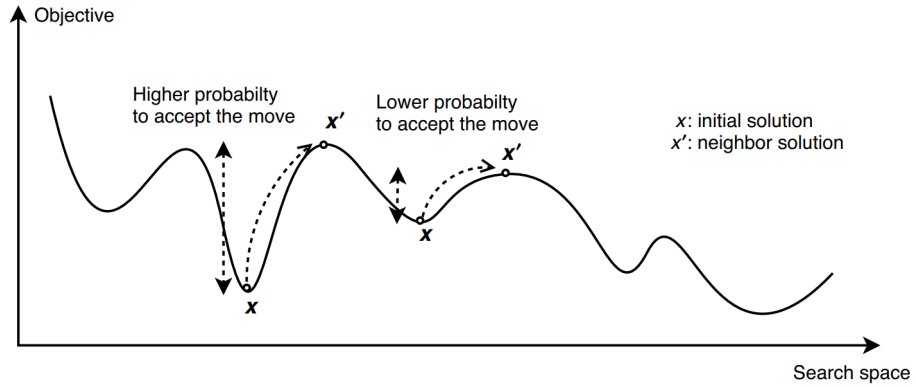


Figura 1.3. Recocido Simulado escapando de un óptimo local. A mayor temperatura, mayor probabilidad de aceptar un peor movimiento. Dada una temperatura, a menor valor de la función objetivo, más significativa será la probabilidad de aceptar el movimiento. Un mejor movimiento es siempre aceptado. Fuente: Talbi (2009)

CAPÍTULO 2

2. METODOLOGÍA

2.1 Búsqueda Local para el VRP

La Búsqueda Local es un método perturbativo e incompleto. La idea es, considerar soluciones una a una y registrar la solución factible de mayor calidad encontrada. El espacio conteniendo todas las posibles soluciones (soluciones factibles y no factibles) a un problema es llamado el espacio de soluciones. La búsqueda local es basada en la observación que modifica una solución dada (factible o no) a una solución nueva y diferente (factible o no). Así, la Búsqueda Local se mueve a través del espacio de soluciones, cada paso o movimiento corresponde a una perturbación de la solución actual. Además analizará a cada solución que encuentra para revisar su factibilidad y registrar la solución factible de mejor calidad encontrada hasta el momento (llamada solución titular).

Dada una solución Γ, el conjunto de soluciones que puede ser obtenido por llevar a cabo una operación σ en Γ es llamada la vecindad $N_\sigma(\Gamma)$ de Γ. La operación σ es desempeñada por aplicar operador de vecindad $\eta_\sigma(\ldots,\Gamma)$ a la solución actual. El tamaño de la vecindad $N_\sigma(\Gamma)$ depende los diferentes parámetros considerados para $\eta_\sigma(\ldots,\Gamma)$. En cada iteración en la Búsqueda Local la vecindad de la solución actual es construida y evaluada. A continuación, se selecciona una de las soluciones vecinas como la actual. Sin embargo, más de un operador de vecindad puede ser usado en la búsqueda local.

2.1.1 Solución Inicial

En la teoría de la perturbación, los métodos de perturbación, como la búsqueda local, parten de una solución inicial dada para empezar a mejorarla. Así, una solución inicial puede venir generada por:

1. **Generación aleatoria:** Consiste en el uso de te una distribución de probabilidad para generar aleatoriamente una ruta factible, de dónde el número de rutas puede también ser aleatorio o simplemente influenciado en el total de la demanda de los clientes y las restricciones de capacidad de los vehículos.
2. **Método de construcción:** Técnicas que nos permiten construir una solución bajo un criterio de selección pero que tiene un costo computacional más alto.
3. **Técnicas mixtas:** Consiste en mejorar una solución a partir de otra encontrada por un método distinto.

En primera instancia, una de las interrogantes aquí será si la solución inicial generada es factible o bien contradice algunas de las condiciones del modelo, por ende es importante tener en cuenta que se reduzca el número de infracciones posibles.

Algoritmo del vecino más cercano

Una solución inicial mediante el método de construcción puede darse mediante el algoritmo del Vecino más cercano, o más conocido en inglés como Nearest Neighbor (NN). Esta solución inicial prioriza encontrar la elección de distancia mínima a partir de un nodo seleccionado, que generalmente corresponderá al nodo del depósito. Consecuentemente, el algoritmo busca el nodo más cercano (vecino más cercano), para colocarlo como siguiente elemento de la ruta,

y así consecutivamente. En Redi et al. (2020), construyen una solución inicial a partir del algoritmo del vecino más cercano donde la metaheurística es el recocido simulado.

2.2 Vecindades

Groër et al. (2010) resumen en una librería escrita en C/C++ los operadores más comunes de búsqueda local para este tipo de problema. De igual forma, Massen (2013); Bräysy and Gendreau (2005) recogen los principales métodos como siguen:

2.2.1 Operador de reubicación

Intuitivamente, este operador conlleva reubicar un nodo de una ruta a otra. Formalmente, dada Γ una solución factible, el operador $\eta_{reloc}\,(i, r_1, p, r_2\,)$, con $(r_1, r_2 \in \Gamma)$, es aquel que toma al nodo i visitado en r_1 y lo remueve de r_1 para reinsertarlo en la posición p en la ruta r_2 y $1 \leqslant p \leqslant |r_2| + 1$.

2.2.2 Operador Swap

Éste operador, puede ser visto como el operador 2.2.1 aplicado dos veces donde los clientes son añadidos en su posición de la contra-parte en la ruta; o bien, como el intercambio de dos clientes de rutas diferentes. Formalmente, dado $r_1, r_2 \in \Gamma$ El operador swap $\eta_{\text{swap}}\ (i, r_1, j, r_2, \Gamma)$,toma dos clientes i y j visitados en las rutas r_1 y r_2 $(r_1 \neq r_2, i \in r_1, j \in r_2)$ y los intercambia. El tamaño de $N_{swap}(\Gamma)$ es determinado por las diferentes rutas y clientes considerados para η_{swap}.

2.2.3 Operador k-opt

Consiste en liberar k aristas en la cual los puntos finales están en las posiciones $< po, p1, ..., p_k >$ en la ruta. Dado $r_1 \in \Gamma, A_1 \cap A_2 = \varnothing$, el operador k-opt $\eta_{k-opt}(r_1, A_1, A_2, \Gamma)$ particiona una ruta r_1 en $k + 1$ segmentos por liberar todos los carcos en A_1 de r_1 con

$\left(A_1 \subseteq \bigcup_{i=0}^{|r|}\{(r[j], r[j+1])\}\right)$ y reconectando los segmentos resultantes usando los arcos en A_2 $\left(A_2 \subseteq \bigcup_{j=0}^{|r+1|}\{\left(\delta^-_{r[j]} \cap \bigcup_{l=0}^{|r+1|}\delta^+_{r[l]}\right)\}\right)$ Note que algunos de los segmentos de la ruta podrían ser reservados en la ruta resultante.

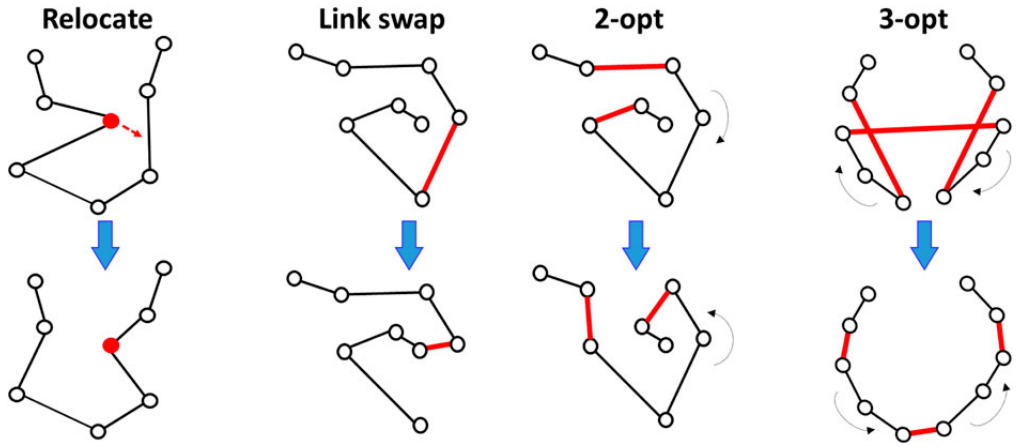


Figura 2.1. Operadores de búsqueda local aplicados a un TSP. Fuente: Mariescu-Istodor and Fränti (2021)

El operador más popular de todos estos, es el $2-opt$, propuesto por Croes (1958). La variante generalizada, denominada $k-opt$ (también conocida como heurística de Lin-Kernighan) propuesta por Lin (1973). El $2-opt$ selecciona dos enlaces y redirige los enlaces entre sus nodos, mientras que el $k-opt$ implica k enlaces a utilizar para lo cual las modificaciones se hacen más complejas en la solución. En Sengupta et al. (2019) se muestra la eficiencia de los operadores en distintas instancias del problema del agente viajero, allí el $2-opt$ muestra una eficiencia del 57% para una serie de instancias de problemas del tipo TSP (bucle cerrado).

2.3 Levantamiento de información

Para el problema del agente viajero, TSP, utilizaremos las instancias previstas por la Universidad de Waterloo. VLSI TSP Collection y National TSP Collection son las instancias a

trabajar para los análisis iniciales de efectividad de los algoritmos involucrados.

Algunas de estas instancias las encontramos en 2.2 2.3

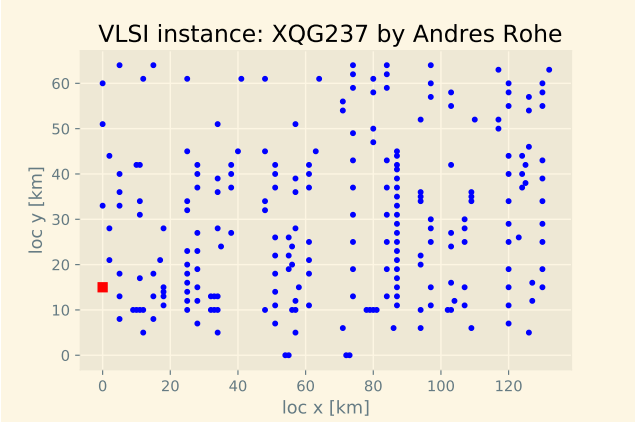


Figura 2.2. Instancia xqg237 de 237 nodos para el Problema del Agente Viajero (TSP) propuesto por Andres Rohe, y estudiado en Forschungsinstitut für Diskrete Mathematik, Universität Bonn. Depósito o nodo inicial de color rojo, demás nodos de color azul

Por otro lado, para el problema de ruteo vehicular capacitado, tenemos las instancias de Augerat, un conjunto de tres clases A, B y P. Dónde las clases A y B corresponden a datos agrupados y no agrupados, mientras el conjunto P corresponde a aquella clase más dinámica respecto a las capacidades de carga. Uchoa et al. (2017) son los autores de la página web (repositorio) *Capacitated Vehicle Routing Problem Library* , de la cuál los datos son extraídos. En la figura 2.4 encontramos un ejemplo de los datos a trabajar.

2.4 Softwares

Las herramientas de trabajo para el presente proyecto son:

1. Python: un lenguaje de programación interpretado, orientado a objetos de alto nivel y con una semántica dinámica. Lin et al. (2022) establece una introducción al lenguaje para principiantes en ingeniería y ciencia. Es actualmente uno de los lenguajes más populares, una de las razones debido a su fácil lectura de código y su versatilidad para

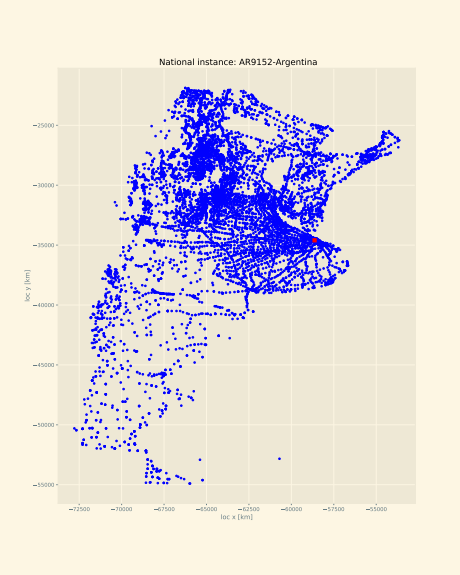


Figura 2.3. Instancia ar9152 del Problema del Agente Viajero (TSP) de 9152 localizaciones. Derivado de datos de la Agencia Nacional de Imágenes y Cartografía

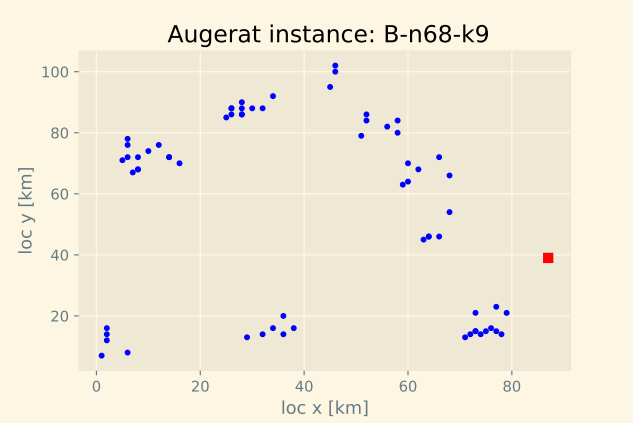


Figura 2.4. Instancia de Augerat B-n68-k9, el depósito o nodo inicial de color rojo, demás nodos de colores azules.

diversos campos de uso: estadística, ciencia de datos, aprendizaje máquina, inteligencia artificial, computación en paralelo, entre otros. Es un lenguaje de licencia libre.

2. MATLAB: es un sistema de cómputo numérico y junto a su lenguaje propio de programación integran computación, visualización y programación en un entorno fácil de usar donde los problemas y las soluciones se expresan en notación matemática

familiar. Es un ambiente de alto desempeño. Lopez (2014); Venkataraman (2009) presentan técnicas de optimización y matemática aplicada desarrolladas en MATLAB. Es el lenguaje más popular para optimización y matemática, y problemas de ingeniería. Es un software privado.

3. Spreadsheet Solver (Excel): Erdoğan (2017) introduce un solucionador de código abierto para los problemas de ruteo vehicular denominado Spreadsheet Solver por su autor. En el mismo artículo, se encuentra que el solucionador es capaz de resolver instancias de VRP capacitado y de VRP con restricciones de distancia con hasta 200 clientes dentro de 1 hora de tiempo de CPU.

2.5 Fases del proyecto

2.5.1 Fase TSP

En esta fase, se analizan las instancias necesarias para trabajar y construir un algoritmo que lo resuelva. Se crea un repositorio público en Github con las debidas instancias a utilizar para proceder a acceder a ellas en la nube desde cualquier computador. Se construyen funciones que nos permitan calcular una matriz o cuadro de datos de las distancias de un nodo respecto a otro. Se estudian las posibles funciones básicas de construcción de solución inicial como el uso de Nearest Neighbor, y algoritmos de búsqueda local como 2-opt.

2.5.2 Fase RS TSP

Aquí se introduce el algoritmo de recocido simulado (RS) para resolver instancias del problema del agente viajero. Se utiliza el operador 2-opt en la búsqueda local, y el método de construcción de una solución inicial es la heurística del vecino más cercano. Se analizan los tiempos de ejecución respecto número de nodos de las instancias, además la eficiencia del algoritmo respecto a las soluciones óptimas.

2.5.3 Fase RS CVRP

Dadas las funciones construidas en las fases anteriores, éstas son pulidas para la extracción de los datos, la creación de la matriz de distancias o cuadros de datos. Además, las funciones de los algoritmos antes programados son extendidas para el problema de CVRP puesto que ahora se trabaja con la condición de capacidad y número de rutas. La función objetivo ahora agrega penalizaciones para las nuevas rutas obtenidas en cada iteración en caso estas sobrepasen los límites de carga establecidos por los datos.

2.6 El algoritmo del recocido simulado

Para el presente trabajo, los pasos generales que se siguen durante la ejecución del algoritmo viene dado por algorithm1. En síntesis, el algoritmo perturba una solución actual denominada $\bar{x}$, como parte de la búsqueda local, y la mejora mediante la búsqueda intensiva de alguno de los operadores antes descritos para obtener una nueva solución x^*. De esta forma, cada que exista una mejora notable en minimizar la función objetivo, el algoritmo guardará la mejor solución x y no la desechará en el resto de la ejecución hasta que aparezca una mejor. Por otro lado, tampoco descarta inmediatamente del todo la nueva solución, sino que puede tender a aceptarla teniendo en cuenta que medida que la temperatura decrece menos probable será de aceptar una peor solución que la actual. Con esto, el algoritmo tratará de salir de un óptimo local y dirigirse lo más posible al óptimo global.

2.7 Estructura de Datos de los lenguajes de programación Python y MATLAB

En Python, las estructuras de los datos son muy variadas que nos ofrece al menos tres opciones para almacenar datos matriciales 2-dimensional: pandas.DataFrame, numpy.array, lista. Así como también, la librería SciPy (escrita en Python), a diferencia de NumPy (escrita

Algorithm 1 Pseudo-código para el algoritmo de Recocido Simulado

Datos: x_0: solución inicial, f: función objetivo, T_0: Temperatura inicial, Tf: Temperatura final, α: coeficiente de decaimiento.

Resultado: $x, f(x)$

1 $\bar{x} \leftarrow x_0; x \leftarrow x_0; T \leftarrow T_0$

while $T > T_f$ **do**

2 **for** $k : 1 \rightarrow kmax$ **do**

3 $x^* \leftarrow N(\bar{x})$

if $f(x^*) < f(x)$ **then**

4 $x \leftarrow x^*$

5 **else**

6 **if** $e^{-\frac{f(x^*)-f(\bar{x})}{T}} > \mathfrak{U}(0,1)$ **then**

7 $\bar{x} \leftarrow x^*$

8 **end**

9 **end**

10 **end**

11 $T = T \times \alpha$

12 **end**

en C), nos ofrece más funcionalidades como la posibilidad de calcular la matriz de distancias sin necesidad de iterar sobre los datos. Por otro lado, el ambiente en MATLAB es distinto, las estructuras tienen una sintaxis e invocación más sencilla y menos diversa.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

Un análisis del desempeño de la metodología propuesta se presenta a continuación, con el objetivo de tener los fundamentos para evaluar la factibilidad de la metaheurística descrita. Para tal fin, será importante observar el tiempo de ejecución del algoritmo para las instancias resueltas, y la calidad de la solución obtenida. No obstante, dado que trabajamos con un algoritmo estocástico, distintas soluciones iniciales nos pueden llevar a una misma solución o bien una misma solución de partida nos puede conducir a distintas soluciones finales, esto debido a las reglas probabilísticas por las cuales se rige el modelo matemático. Así, es necesario tomar una muestra cuyo tamaño va acorde al tiempo de ejecución del algoritmo entre las instancias a comparar, y se analiza tanto el tiempo promedio de ejecución y el puntaje promedio obtenido por cada instancia a resolver. Una muestra de 10 soluciones finales por cada instancia, se establece para hacer los análisis. Puesto que el tiempo total conjunto no superaría los 5 minutos de ejecución, y de éstas tomamos la solución que tengo el menor valor de función objetivo (menor distancia recorrida), se llama *mejor solución* o bien, *menor solución*

Los ensayos computacionales son llevados a cabo en un cómputo de características Intel(R) Core(TM) i7-7700HQ CPU@ 2.80GHz 8.00 GB RAM, cuyo sistema operativo es Windows de 64bit. El uso del lenguaje de programación Matlab se lleva a cabo en su homónimo entorno de desarrollo integrado. Para la primera fase, se analiza también el desempeño del lenguaje de programación Python, para lo cuál se usa Visual Studio Code

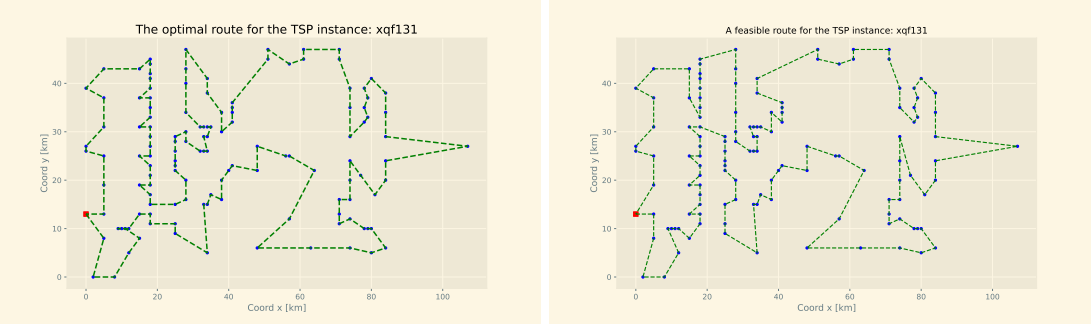
como entorno de desarrollo.

3.1 TSP: Travelling Salesman Problem

Para el TSP, el modelo matemático es codificado tanto en Python y Matlab. En principio, encontrar una solución para las denominadas instancias ”pequeñas”, es decir, un número de nodos o ciudades no mayores a 200, ambos lenguajes logran soluciones aceptables no mayor al 10% respecto al óptimo. En 3.1 se puede apreciar un resultado relativamente aceptable respecto al número de nodos. Para Python, se hace uso de `cdist` de la librería `scipy.spatial.distance`; para MATLAB, se hace uso de `pdist2` y `round(D, 0)` para calcular la matriz de Distancia D de un conjunto de datos, y a su vez redondear los datos a enteros.

En principio, los arreglos provistos por Numpy en Python son la estructura ideal para trabajar, superando con creces a los DataFrames de Pandas, pero sin lograr superar la eficiencia del entorno de desarrollo de MATLAB. Obsérvese las tablas 3.1, 3.2, de dónde se aprecia la superioridad en el desempeño de los tiempos promedios, e incluso trabajando con las mismas distribuciones uniformes para perturbar las soluciones, Python no logra procesar el algoritmo 2-opt respecto a su rival. La superioridad de MATLAB es contundente, Python resulta ser ineficiente para resolver problemas del agente viajero. De esta forma, se descarta el uso de Python para las siguientes simulaciones puesto que no generaría tiempos tolerables. Si bien, estos son resultados preliminares, con la expectativa de mejorarlos.

La búsqueda de soluciones estaría ligada no sólo al tipo de heurística para el que apoyemos nuestro algoritmo, generar una buena vecindad nos conduciría a poder obtener una mejor solución. Sin embargo, dado los tiempos en los que el problema es resuelto, se puede aumentar el número de iteraciones para intentar obtener mejores soluciones. De esta



(a) Ruta óptima de la instancia xqf131 con distancia total de 564.

(b) Solución del Recocido Simulado, con una distancia total de 575

Figura 3.1. Contraste entre una solución óptima y una solución aproximada. Fuente: Creación propia.

Tabla 3.1. Una comparativa de la eficiencia del lenguaje de programación Python y MATLAB. Fuente: Creación propia.

k=5

Instancia	n	Óptimo	Tiempo Python (sg)	Tiempo Matlab (sg)
XQF	131	564	7.9	0.1
XQG	237	1019	29	0.2
PMA	343	1368	16.3	0.5
PKA	379	1332	19.5	0.7
BCL	380	1621	19.2	0.6
PBK	411	1343	22.3	0.8
PBN	423	1365	23.8	0.8
PBM	436	1443	24.9	0.9
XQL	662	2513	60.0	2.1

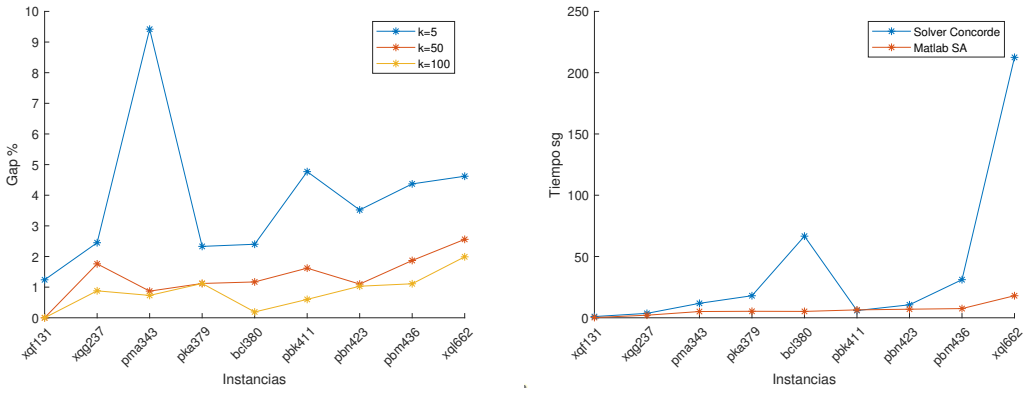
forma, la figura 3.2a, podemos observar una mejora plausible mientras que el aumento del tiempo transcurrido durante el algoritmo presenta una aumento tolerable que se encuentra en A.2, A.3, A.4, establecidas en el ANEXO A.

Tabla 3.2. Una comparativa de la eficiencia del lenguaje de programación Python y MATLAB del GAP. Fuente: Creación propia.

k=5			Python	Matlab
Instancia	n	Óptimo	GAP(%)	GAP(%)
XQF	131	564	2.48	1.24
XQG	237	1019	4.39	2.45
PMA	343	1368	6.06	9.42
PKA	379	1332	6.23	2.33
BCL	380	1621	7.53	2.40
PBK	411	1343	9.61	4.77
PBN	423	1365	7.98	3.52
PBM	436	1443	7.76	4.37
XQL	662	2513	2.38	4.62

El aumento en el lapso de ejecución resulta tolerable en lo expuesto en A.4, ya que el mínimo encontrado no supera 2.5% respecto al óptimo (figura 3.2a). En el trasfondo, es posible encontrar soluciones cuya calidad no sea superior al 2.5% respecto al óptimo global, cuando el número de nodos sea menor a 662.

La estructura en la que están distribuidas las ciudades pretende jugar un rol importante. Se observa que la instancia pbn423 posee una mejor aproximación respecto a su antecesora pbk411 (411 ciudades), y su sucesora pbm436 (436 ciudades). Esto augura la idea de que el tamaño de nodos para un problema TSP no es el único factor que influya en alcanzar el óptimo global, planteando la hipótesis sobre la distribución de los datos siendo un factor relevante a tomar en cuenta. Esto cobra fuerza al observar el comportamiento de la instancia bcl380, en las tablas A.1, A.2, A.3, A.4, en dónde bcl380 muestra un menor tiempo de ejecución, un comportamiento similar al de pbn423 pero respecto al lapso promedio transcurrido.



(a) Efecto de incrementar k

(b) Concorde vs Matlab

Figura 3.2. Comparativas de los resultados obtenidos en TSP. Fuente: Creación propia

El algoritmo de RC elaborado en Matlab, presenta tiempos de ejecución menores a al tiempo que el solver Concorde lleva a cabo al resolver las instancias mediante métodos exactos (branch and bound, branch and cut). En figura 3.2 Se observa que para la instancia de mayor tamaño, a Concorde le toma al menos 10 veces más de lo que el recocido simulado con 100 iteraciones corre. Sin embargo, para la instancia de 662 nodos, y cualquier instancia con nodos mayores a 600 en el conjunto de datos del VLSI, el tiempo de ejecución para el Concorde será al menos de 3.5 minutos lo cuál dificultará la búsqueda de soluciones para instancias de ciudades superior a 700. Y es aquí dónde la metaheurística toma fuerza, pierde levemente calidad, pero gana en el tiempo de ejecución del algoritmo. De aquí tenemos dos observaciones:

- En la figura 3.2a, vemos que al aumentar el número de iteraciones, se consigue conseguir mejores soluciones.
- Para una instancia de mayor tamaño, DKG813, el software Concorde lleva a cabo un tiempo de solución de 45 minutos, el puntaje óptimo tiene un valor de 3199. Esto, para una empresa logística resultaría ser un tema preocupante, ya que no siempre se

dispondría de aquella cantidad de tiempo, especialmente si nuevos problemas surgen periódicamente.

- Mediante el algoritmo del Vecino más cercano, se provee al algoritmo una solución de mejor calidad que perturba y mejora en cada etapa del enfriamiento. Se obtiene que, una solución con un GAP de hasta el 1.4% con un tiempo promedio de 38 segundos. El puntaje del Recocido Simulado, es de 3244, alrededor de 50 puntos de diferencias. A partir de aquí acercarse al óptimo no necesariamente conllevaría aumentar el número de iteraciones desmedidamente, se podría plantear el uso de diferentes heurísticas de mejora.

- Finalmente, en la figura 3.2b, cabe destacar que los recursos con los que cuente una empresa puede ser insuficiente respecto a la demanda exigida por el software tradicional, como método exacto. De esta forma, se mejora el tiempo de ejecución, pero mejorar la calidad resulta un reto a partir de este tamaño de nodos superior a 700.

En la figura 3.3 se corrobora la teoría probabilística en la que yace el recocido simulado. Como se observa, el algoritmo no necesariamente obtiene una mejor solución global en la siguiente iteración, puesto que inicialmente logra aceptar peores soluciones a la actual (cuando la temperatura inicial es alta); sin embargo a medida que la temperatura disminuye la tendencia a aceptar una peor solución fluctúa, el criterio de aceptación es menos flexible en las últimas etapas. Además, como se observa el algoritmo mantiene guarda siempre la mejor solución y sólo en caso de haber una mejor la acepta. Es la solución actual aquella varía en el enfriamiento, con el objetivo de escapar de óptimos locales. Esto muestra que el algoritmo está listo para enfrentarse a un problema *más difícil*, en el sentido de aumentar restricciones y variables que se presentan en el mundo real.

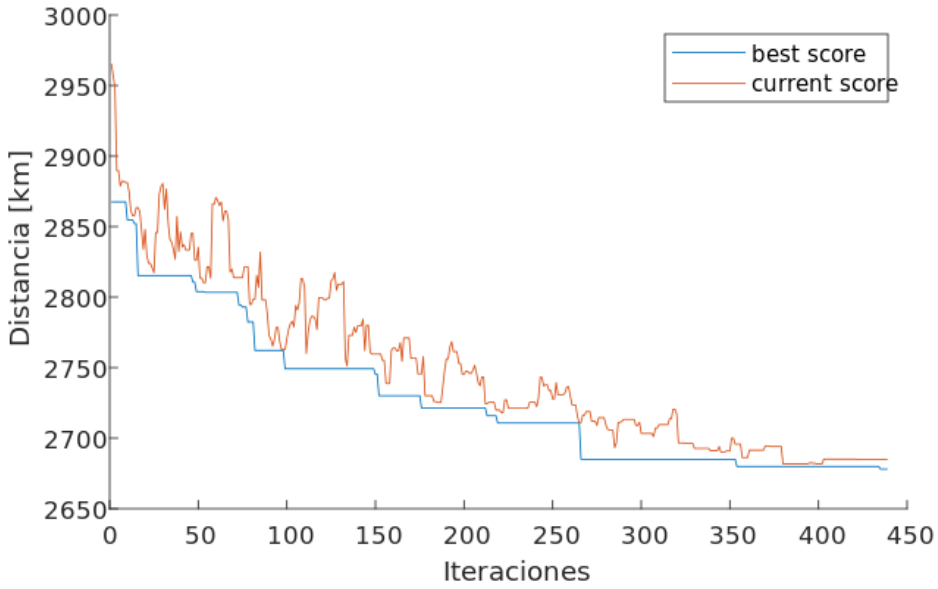


Figura 3.3. Contraste del score obtenido durante las iteraciones del algoritmo recocido simulado para la instancia XQL662. Fuente: Creación propia

3.2 CVRP: Capacitated Vehicle Routing Problem

El algoritmo ha lo grado resolver de forma óptima algunas instancias de clase A del benchmark Augerat, las cuales podemos encontrar en la tabla 3.3 y tiempos de ejecución notablemente bajos, puesto que las primeras instancias de la clase A no requieren mayor dificultad para alcanzar el óptimo. A medida que el algoritmo se enfrente a instancias más grandes, o datos cuya distribución sea más compleja, se necesita mayor tiempo de cómputo para lograr una calidad de solución relativamente aceptable. Lo cierto es, en general se aceptan soluciones hasta el 10% respecto al óptimo, aquí se ha logrado incluso alcanzar el óptimo global en problemas con un número de nodos superior a 50 y hasta 7 vehículos, cómo se observa en tabla 3.3. Los resultados de la tabla B.2, y B.1, indican que el modelo no sólo trabaja bien para datos agrupados, sino también efectivamente en aquellos que no están agrupados. Si bien, para el caso B.2 se tiene una desviación estándar menor respecto a B.1

en los GAPs de las mejores soluciones obtenidas, por lo que los resultados en B.2 tienden a estar más ajustados.

El desempeño mostrado hasta el momento por el algoritmo resulta ser confiable, y pretender resolver el problema de ruteo vehicular que es *más difícil* que el TSP. Por lo que es de esperar, que el tamaño de las instancias sean mucho menor. Se espera además que el Recocido Simulado logre buenas soluciones, en tiempos no superiores a 10 segundos para que una empresa pueda fácilmente tomar decisiones apoyada en datos.

Tabla 3.3. Instancias de Augerat clase A, clientes no mayores a 50, resueltas por RC. Fuente: Creación propia.

capacidad=100

Instancia	n	Vehículos	Óptimo	Media	Mínimo	GAP	Tiempo (sg)
A-n32-k5	32	5	784	811	784	0	0.2
A-n33-k6	33	6	742	744	742	0	0.2
A-n34-k5	34	5	778	787	778	0	1
A-n36-k5	36	5	779	823	805	3.34	2.0
A-n37-k5	37	5	669	675	669	0	2.3
A-n37-k6	37	6	949	991	958	0.95	2.4
A-n38-k5	38	5	730	738	730	0	2.4
A-n39-k5	39	5	822	844	825	0.36	2.4
A-n39-k6	39	6	831	843	834	0.36	2.37
A-n44-k6	44	6	937	981	958	2.2	2.5
A-n45-k6	45	6	944	984	968	2.5	2.4
A-n45-k7	45	7	1146	1210	1185	3.4	2.6
A-n46-k7	46	7	914	958	914	0	2.6
A-n48-k7	48	7	1073	1122	1073	0	2.6

La clase P de las instancias de Augerat, expuestos en 3.5, clase que tiene datos más

Tabla 3.4. Instancias de Augerat clase B, clientes mayores a 50, resueltas por RC. Fuente: Creación propia.

capacidad=100

Instancia	n	Vehículos	Óptimo	Media	Mínimo	GAP	Tiempo (sg)
B-n50-k7	50	7	741	748	741	0	5.1
B-n50-k8	50	8	1312	1356	1333	1.6	5.1
B-n51-k7	51	7	1032	1043	1034	0.2	4.9
B-n52-k7	52	7	747	748	747	0	5.0
B-n56-k7	56	7	707	723	715	1.1	5.3
B-n57-k9	57	9	1598	1651	1620	1.3	5.4
B-n63-k10	63	10	1496	1596	1546	3.3	5.4
B-n64-k9	64	9	861	913	888	3.3	5.5
B-n66-k9	66	9	1316	1349	1329	0.99	5.7
B-n67-k10	67	10	1032	1088	1044	1.2	6.2
B-n68-k9	68	9	1272	1332	1298	2.0	5.81
B-n78-k10	78	10	1221	1298	1258	3.0	6.2

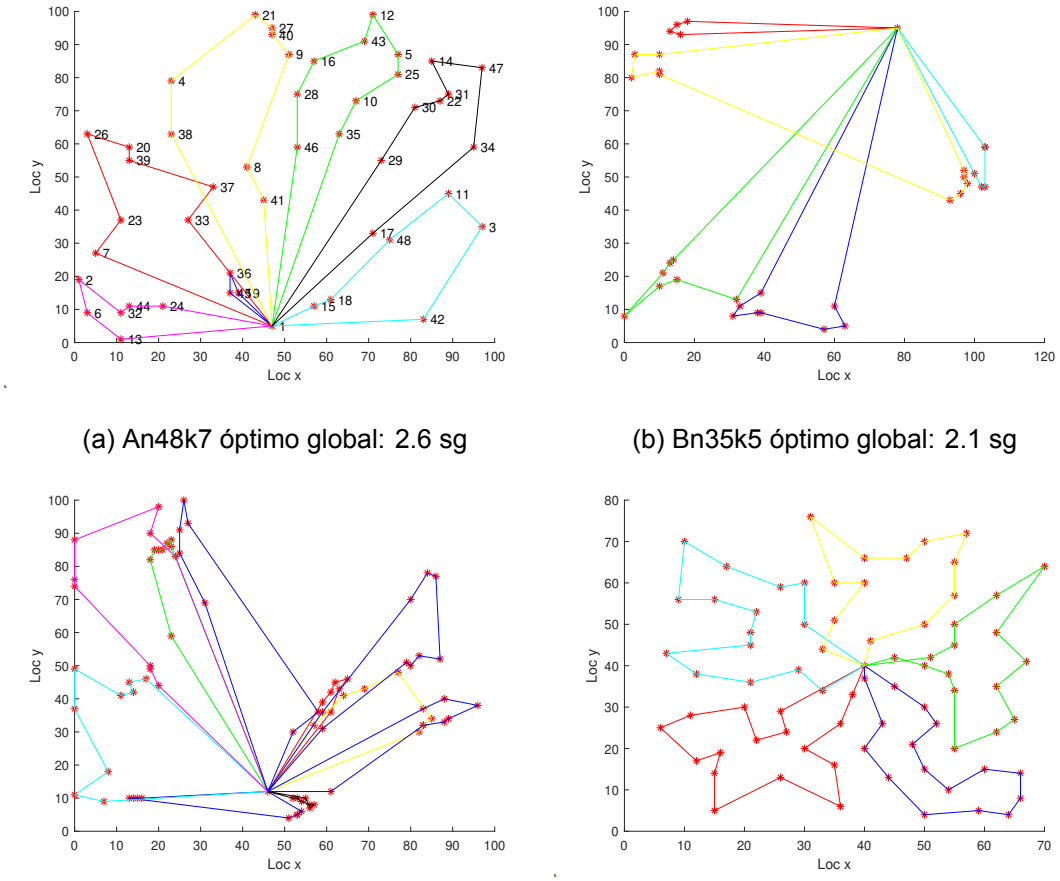
diversos en cuanto a vehículos, nodos, y limitaciones en la capacidad, muestra resultados similares los obtenidos por los demás conjuntos de datos. Se encontraron muy buenas soluciones en instancias con un número de nodos menor a 50 en tiempo sorprendentemente cortos, demostrando que el número de vehículos, la capacidad de carga y distancias medias entre nodos son un factor importante a tener en cuenta respecto a qué tan complejo se vuelve resolver un VRP. Por eso, lo que fácilmente se consigue en menos de 1 segundo en P-n45-k5 con un número de iteraciones fijo, no podría conseguirse en las instancias A y B, puesto que necesitan de más iteraciones en el algoritmo para lograr un resultado aceptable.

En figura 3.4, podemos observar algunas soluciones con sus tiempos de resolución.

Tabla 3.5. Instancias de Augerat clase P, resueltas por RC. Fuente: Creación propia.

Instancia	n	Vehículos	Cap	Óptimo	Media	Mínimo	GAP	Tiempo (sg)
P-n16-k8	16	8	35	450	451	450	0	0.2
P-n19-k2	19	2	160	212	218	212	0	0.1
P-n20-k2	20	2	160	216	2018	216	0	0.1
P-n21-k2	21	2	160	211	213	211	0	0.1
P-n23-k8	23	8	40	529	535	529	0	0.6
P-n40-k5	40	5	140	458	470	461	0.6	0.7
P-n45-k5	45	5	150	510	520	515	0.9	0.7
P-n50-k7	50	7	150	554	564	558	0.7	4.5
P-n50-k8	50	8	120	631	649	638	1.1	5.1
P-n50-k10	50	10	100	696	725	710	2.0	5.2
P-n55-k7	55	7	170	568	586	577	1.5	6.0
P-n55-k10	55	10	115	694	716	708	2.0	5.8
P-n65-k10	65	10	130	792	830	815	2.9	5.9
P-n70-k10	70	10	135	827	876	848	2.5	6.3
P-n76-k4	76	4	350	593	619	610	2.8	7.0
P-n76-k5	76	5	280	627	658	637	1.5	6.5
P-n101-k4	101	4	400	681	699	690	1.3	8.0

De esta forma se aprecia la forma de un óptimo global, que no necesariamente implica que un vehículo tenga que dirigirse a un conjunto específico de nodos, sino que puede cruzarse en rutas con otro vehículo como ocurre en 3.4b, al contrario de 3.4a. Además, 3.4c no muestra que no necesariamente todas las rutas deberán ser largas o con un número excesivo de nodos. En 3.4d, vemos rutas más homogéneas. Aunque hay aproximadamente 30 ciudades o clientes de diferencia entre la primera instancia, respecto a las últimas, la



(a) An48k7 óptimo global: 2.6 sg

(b) Bn35k5 óptimo global: 2.1 sg

(c) Bn78k10: 3.3% y 6 sg

(d) Pn76k5: 2.4% y 6.1 sg

Figura 3.4. Comparativas de los resultados obtenidos. Fuente: Creación propia.

complejidad aumenta de tal forma que encontrar un óptimo resulta ser más complejo, requiriendo involucrar otras heurísticas de búsqueda local que nos puedan garantizar rutas más eficientes sin necesidad de aumentar la potencia, que traería consigo tiempos de resolución más altos.

Mediante el uso de Spreadsheet Solver, hacemos una comparativa de la eficiencia del Recocido Simulado y la metaheurística propia del solver informático. Los resultados obtenidos por Spreadsheet Solver son expuestos en la tabla 3.6, contrastándose un empate técnico a nivel de optimización de distancias al ser ambos algoritmos estocásticos. Por otro lado es notable el aumento del tiempo comparado al recocido Simulado. En tabla C.1 se constata la

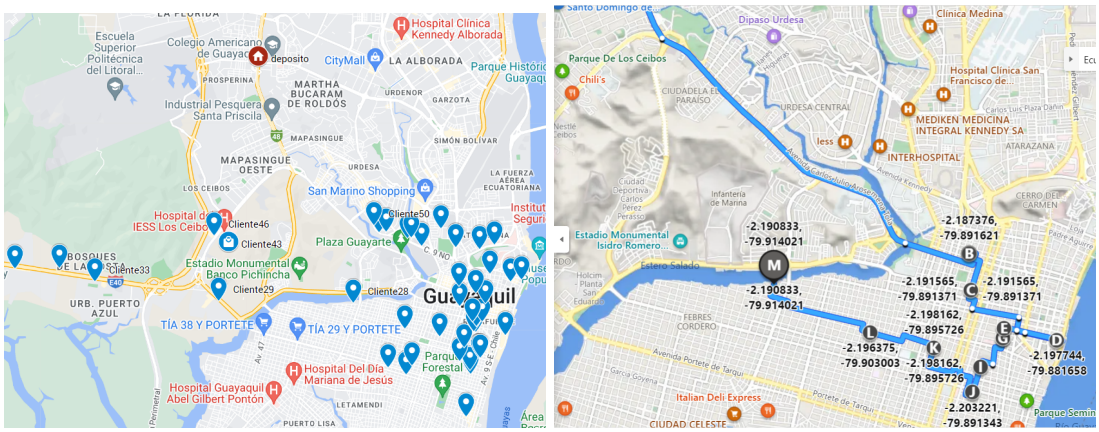
Tabla 3.6. Una comparativa de la eficiencia del algoritmo desarrolado frente a otros softwares. Fuente: Creación propia.

			Matlab: RC		Spreadsheet Solver	
Instancia	Cap	Óptimo	Solución	Tiempo (sg)	Solución	Tiempo (sg)
A-n48-k7	100	1073	1073	2.6	1073	600
B-n66-k9	100	1316	1329	5.7	1325	600
P-n65-k10	130	792	815	6.9	796	600
P-n50-k8	120	631	638	5.1	635	600
P-n101-k4	400	681	690	8	692	600

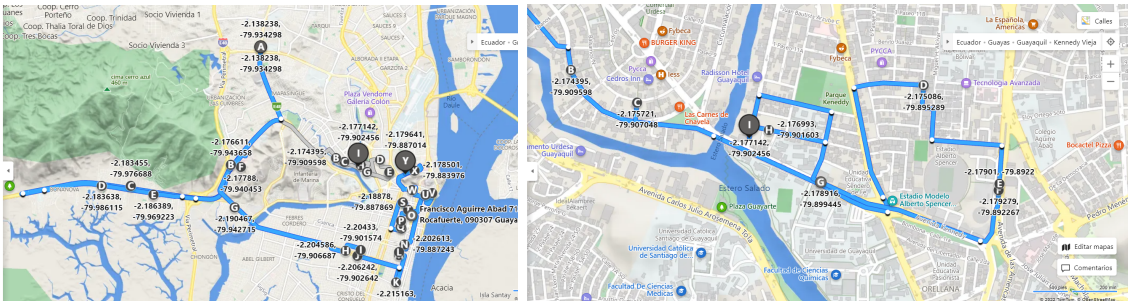
calidad del GAP de la solución encontrada para cada instancia. En general, los resultados son muy buenos para ambos softwares, exceptuándo la limitaciones de tiempo para el solver en Excel.

Aunque los resultados mostrados en las tablas y figuras previas son instancias disponibles en la literatura, Spreadsheet Solver posee acceso a la plataforma de Bing Maps, de modo que es posible utilizar latitudes y longitudes de clientes y depósitos, cálculo de la matriz de distancias mediante el tiempo de conducción que tomen las ubicaciones unas otras, entre otras características. Un resultado es mostrado en figura 3.5 para la ciudad de Guayaquil, con un depósito ubicado en la vía a Daule y la avenida Juan Tanca Marengo; con 50 clientes esparcidos alrededor de vía a la Costa, centro de Guayaquil. El caso cuenta con la disponibilidad de 3 vehículos, cada uno de 1 tonelada; un total a distribuir de 2.6 toneladas de mercancías en las inmediaciones de la ciudad. Este ejemplo, muestra cómo funcionaría lo estudiado previamente para aplicaciones reales en empresas de distribución, ya que no se trabaja con la distancia euclidiana redondeada como en los casos anteriores. El número de unidades a entregar se transforma en peso de mercancía en específico demandada. Como

podemos observar en 3.5d se observa que el camino más corto no necesariamente es el que se puede ejecutar,puesto que las calles no necesariamente son doble vía, o que desde las avenidas principales se permita girar hacia una avenida en particular, aumentando las complejidad del caso en término de distancias recorridas.



(a) Conjunto de clientes. (b) Ruta 1.



(c) Ruta 2. (d) Ruta 3.

Figura 3.5. Comparativas de los resultados obtenidos. Fuente: Creación propia.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

En esta seccíon se presentan las conclusiones y recomendaciones obtenidas de los resultados de este trabajo, donde se utilizó la metaheurística de recocido simulado apoyada por heurística de búsqueda local para la resolución del Problema de Ruteo Vehicular Capacitado (CVRP, por sus siglas en inglés), y su vez para el Problema del Agente Viajero (TSP, por sus siglas en inglés)

4.1 Conclusiones

El propósito de este trabajo fue optimizar el problema de ruteo mediante la metaheurística del recocido simulado. Con base en los resultados previos se constata que se halló ”buenas soluciones” respecto al tiempo de ejecución en instancias de datos homogéneos y no homogéneos, lo que permite establecer que el modelo meta-heurístico de recocido simulado optimiza rutas de transporte acortando considerablemente las distancia sin el uso excesivo de recursos computacionales y tiempos de computo razonables, por ende, rebajando los costos de transporte para permitir a las empresas logísticas ser más competitivas. El modelo matemático trabajo puede ser usado como herramienta de apoyo en la toma de decisiones en empresas dedicadas a la distribución de mercancías

Los lapsos de ejecución para instancias del TSP con un tamaño menor 700 de las mejores soluciones, no exceden el minuto de procesamiento en el lenguaje de programación Matlab. La implementación de la metaheurística en el problema del agente viajero conduce

a tiempos de solución admisibles y ”buenas soluciones” respecto a softwares tradicionales, resultando ser confiable al momento de abordar instancias del TSP.

4.2 Recomendaciones

Aunque Matlab presenta mejoras significativas, habría la posibilidad de mejorar el algoritmo de Python, descargando librería py2opt. Este solver asegura resolver el TSP mediante la función RouteFinder del paquete py2opt.routefinder, entre los parámetros que acepta está la matriz de distancia, los nombres de las ciudades, y un número de iteraciones k que de forma predeterminada es 5. Dado que la librería es de licencia abierta, podemos encontrarla en el repositorio de Github: https://github.com/pdrm83/py2opt/tree/master/py2opt. Para poder usarla, dentro del Recocido Simulado, es necesario entrar al código fuente de las funciones de py2opt y retirar la impresión del puntaje de la distancia en la ventana de comando generado por cada iteración, ya que ralentizaría la ejecución del RC. No obstante, la misma librería establece que para un TSP con nodos de 120, la solución podrá darse en un lapso menor a 5 segundos. En caso de que esta librería pueda mejorar la eficiencia del algoritmo para Python, sería una contribución para aquellos que manejan dicho lenguaje de programación.

Una mejora para el algoritmo del CVRP, una solución inicial basada en métodos de agrupamiento de datos para la clase P (de datos agrupado), podría generar soluciones de calidad sin tanto esfuerzo computacional. Prueba del algoritmo mediante otras instancias disponibles en http://vrp.atd-lab.inf.puc-rio.br/index.php/en/. Por otro lado, la implementación de nuevos métodos de búsqueda local para comparar con el algoritmo de 2-opt, tanto en la calidad y tiempo de ejecución.

La complejidad del VRP se muestra notable a medida que el número de nodos o el

número de vehículos incrementa, de modo que se recomienda incorporar al modelo heurísticas más variadas que permitan encontrar soluciones más cercanas al óptimos, o disminuir los tiempos de solución alterar significamente la calidad. No se recomienda el uso del lenguaje Python para resolver un VRP, puesto que los tiempos no de ejecución no serían tolerables.

BIBLIOGRAFÍA

Aaronson, S. (2013). Why philosophers should care about computational complexity. *Computability: Turing, Gödel, Church, and Beyond*, 261:327.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (1995). *Finding cuts in the TSP (A preliminary report)*, volume 95. Citeseer.

Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. (2011). *The Traveling Salesman Problem*. Princeton University Press, Princeton.

Barcia Kleber, Gutiérrez Ronald, G. V. (2019). Vehicle routing with time windows applying the clarke and wright algorithm for a medical supplies company. In *17 th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Industry, Innovation and Infrastructure for Sustainable Cities and Communities"*.

Berger, B. and Leighton, T. (1998). Protein folding in the hydrophobic-hydrophilic (hp) is np-complete. In *Proceedings of the second annual international conference on Computational molecular biology*, pages 30–39.

Boltzmann, L. (1868). Studien uber das gleichgewicht der lebenden kraft (studies on the balance of living force between moving material points). *Wissenschafiliche Abhandlungen*, 1:49–96.

Bräysy, O. and Gendreau, M. (2005). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118.

Christofides, N., Mingozzi, A., and Toth, P. (1981a). Exact algorithms for the vehicle routing

problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282.

Christofides, N., Mingozzi, A., and Toth, P. (1981b). State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2):145–164.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581.

Cook, S. (2000). The p versus np problem. *Clay Mathematics Institute*, 2.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158.

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

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

De, D., Kumarasubramanian, A., and Venkatesan, R. (2007). Inversion attacks on secure hash functions using sat solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 377–382. Springer.

Erdoğan, G. (2017). An open source spreadsheet solver for vehicle routing problems. *Computers & operations research*, 84:62–72.

Fortnow, L. (2021). Fifty years of p vs. np and the possibility of the impossible. *Communications of the ACM*, 65(1):76–85.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549.

Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206.

Glover, F. (1990). Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32.

Groër, C., Golden, B., and Wasil, E. (2010). A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2(2):79–101.

Hasle, G., Lie, K.-A., and Quak, E. (2007). *Geometric modelling, numerical simulation, and optimization*. Springer.

Horie, S. and Watanabe, O. (1997). Hard instance generation for sat. In *International Symposium on Algorithms and Computation*, pages 22–31. Springer.

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

Laporte, G. (1992a). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247.

Laporte, G. (1992b). The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59(3):345–358.

Laporte, G. and Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. volume 132, pages 147–184. Elsevier.

Lenstra, J. K. and Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

Lin, J. W.-B., Aizenman, H., Espinel, E. M. C., Gunnerson, K., and Liu, J. (2022). *An Introduction to Python Programming for Scientists and Engineers*. Cambridge University Press.

Lin, S. (1973). An efficient heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516.

Lopez, C. (2014). *MATLAB optimization techniques*. Apress.

Mariescu-Istodor, R. and Fränti, P. (2021). Solving the large-scale tsp problem in 1 h: Santa claus challenge 2020. *Frontiers in Robotics and AI*, 8.

Massacci, F. and Marraro, L. (2000). Logical cryptanalysis as a sat problem. *Journal of Automated Reasoning*, 24(1):165–203.

Massen, F. (2013). *Optimization approachs for the vehicle routing problem with black box feasibility*. PhD thesis, UCL-Université Catholique de Louvain.

Mor, A. and Speranza, M. G. (2022). Vehicle routing problems over time: a survey. *Annals of Operations Research*, pages 1–21.

Paolo Toth, D. V. e. (2014). *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. MOS-Siam Series on Optimization. Society for Industrial and Applied Mathematics, 2 edition.

Rash, E. and Kempf, K. (2012). Product line design and scheduling at intel. *Interfaces*, 42(5):425–436.

Redi, A. A. N. P., Maula, F. R., Kumari, F., Syaveyenda, N. U., Ruswandi, N., Khasanah, A. U., Kurniawan, A. C., et al. (2020). Simulated annealing algorithm for solving the capacitated vehicle routing problem: a case study of pharmaceutical distribution. *Jurnal Sistem dan Manajemen Industri*, 4(1):41–49.

Sanabria Quiñonez, F. Y. (2018). Estudio comparativo de algoritmos basados en

metaheurísticas aplicados a la solución del problema de ruteo de vehículos con capacidad limitada. Master's thesis, Escuela Superior Politécnica del Litoral.

Sengupta, L., Mariescu-Istodor, R., and Fränti, P. (2019). Which local search operator works best for the open-loop tsp? *Applied Sciences*, 9(19):3985.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. Wiley Series on Parallel and Distributed Computing. Wiley, new edition.

Tan, S.-Y. and Yeh, W.-C. (2021). The vehicle routing problem: State of the art classification and review. *Applied Sciences*, 11(21).

Tasan, A. S. and Gen, M. (2012). A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers Industrial Engineering*, 62(3):755–761. Soft Computing for Management Systems.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., and Subramanian, A. (2017). New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858.

Venkataraman, P. (2009). *Applied optimization with MATLAB programming*. John Wiley & Sons.

APÉNDICES

APÉNDICE A

Tabla A.1. Instancias VLSI, del Problema del Agente Viajero, resultas por Recocido Simulado en el lenguaje de programación Python

k=5

Instancia	n	Óptimo	Media	Mínimo	GAP	Tiempo (sg)
XQF131	131	564	583	578	2.48	7.9
XQG237	237	1019	1077	1066	4.39	29
PMA343	343	1368	1458	1451	6.06	16.3
PKA379	379	1332	1425	1415	6.23	19.5
BCL380	380	1621	1760	1743	7.53	19.2
PBK411	411	1343	1472	1472	9.61	22.3
PBN423	423	1365	1480	1474	7.98	23.8
PBM436	436	1443	1568	1555	7.76	24.9
XQL662	662	2513	2598	2573	2.38	60.0

Tabla A.2. Instancias resueltas del benchmark VLSI , ejecutadas con $k = 50$ iteraciones. Software: MATLAB. Fuente: Creación propia.

k=5

Instancia	n	Óptimo	Media	Mínimo	GAP	Tiempo (sg)
XQF131	131	564	576	571	1.24	0.1
XQG237	237	1019	1062	1044	2.45	0.2
PMA343	343	1368	1419	1497	9.42	0.5
PKA379	379	1332	1382	1363	2.33	0.7
BCL380	380	1621	1684	1660	2.40	0.6
PBK411	411	1343	1416	1407	4.77	0.8
PBN423	423	1365	1431	1413	3.52	0.8
PBM436	436	1443	1522	1506	4.37	0.9
XQL662	662	2513	2671	2629	4.62	2.1

Tabla A.3. Instancias resueltas del benchmark VLSI , ejecutadas con $k = 50$ iteraciones. Software: MATLAB. Fuente: Creación propia.

k=50

Instancia	n	Óptimo	Media	Mínimo	GAP %	Tiempo (sg)
XQF131	131	564	569	564	0.0	0.2
XQG237	237	1019	1042	1037	1.76	2.5
PMA343	343	1368	1388	1380	0.87	6.4
PKA379	379	1332	1357	1347	1.12	8.2
BCL380	380	1621	1649	1640	1.17	7
PBK411	411	1343	1376	1365	1.62	8.5
PBN423	423	1365	1395	1380	1.1	9.3
PBM436	436	1443	1480	1470	1.87	10.1
XQL662	662	2513	2586	2577	2.56	24.7

Tabla A.4. Instancias resultas del benchmark VLSI, ejecutadas con $k = 100$ iteraciones. Software: MATLAB. Fuente: Creación propia.

k=100

Instancia	n	Óptimal	Avg	Best	GAP (%)	Time (s)
XQF131	131	564	566	564	0.0	0.2
XQG237	237	1019	1031	1028	0.88	2.1
PMA343	343	1368	1383	1378	0.73	5.1
PKA379	379	1332	1352	1347	1.12	5.3
BCL380	380	1621	1646	1624	0.19	5.2
PBK411	411	1343	1370	1355	0.60	6.45
PBN423	423	1365	1386	1379	1.03	7.0
PBM436	436	1443	1470	1459	1.11	7.5
XQL662	662	2513	2573	2563	1.99	18

APÉNDICE B

Tabla B.1. Instancias de Augerat clase A, clientes no mayores a 50, resueltas por RC

capacidad=100

Instancia	n	Vehículos	Óptimo	Media	Mínimo	GAP	Tiempo (sg)
A-n32-k5	32	5	784	811	784	0	0.2
A-n33-k6	33	6	742	744	742	0	0.2
A-n34-k5	34	5	778	787	778	0	1
A-n36-k5	36	5	779	823	805	3.34	2.0
A-n37-k5	37	5	669	675	669	0	2.3
A-n37-k6	37	6	949	991	958	0.95	2.4
A-n38-k5	38	5	730	738	730	0	2.4
A-n39-k5	39	5	822	844	825	0.36	2.4
A-n39-k6	39	6	831	843	834	0.36	2.37
A-n44-k6	44	6	937	981	958	2.2	2.5
A-n45-k6	45	6	944	984	968	2.5	2.4
A-n45-k7	45	7	1146	1210 ca	1185	3.4	2.6
A-n46-k7	46	7	914	958	914	0	2.6
A-n48-k7	48	7	1073	1122	1073	0	2.6
A-n53-k7	53	7	1010	1048	1026	1.58	3.9
A-n54-k7	54	7	1167	1214	1190	1.97	4.1
A-n55-k9	55	9	1073	1106	1082	0.88	5.5
A-n60-k9	60	9	1354	1401	1371	1.26	5.49
A-n61-k9	61	9	1034	1079	1043	0.87	5.51
A-n62-k8	62	8	1288	1357	1323	3.5	5.93
A-n63-k9	63	9	1616	1723	1654	2.35	5.7
A-n63-k10	63	10	1314	1378	1346	2.44	5.9
A-n64-k9	64	9	1401	1465	1429	1.99	5.87
A-n65-k9	65	9	1174	1209	1190	1.36	5.82
A-n69-k9	69	9	1159	1207	1189	2.56	6.00
A-n80-k10	80	10	1763	1863	1813	2.84	12

Tabla B.2. Instancias de Augerat clase B, resueltas por RC

cap=100

Instancia	n	Vehículos	Óptimo	Media	Mínimo	GAP	Tiempo (sg)
B-n31-k5	31	5	672	683	672	0	2.4
B-n34-k5	34	5	788	793	788	0	2.4
B-n35-k5	35	5	955	961	956	0	2.7
B-n38-k6	38	6	805	810	805	0	2.6
B-n39-k5	39	5	549	568	554	0.9	2.3
B-n41-k6	41	6	829	842	835	0.6	2.7
B-n43-k6	43	6	742	746	743	0.1	2.6
B-n44-k7	44	7	909	940	927	1.9	2.5
B-n45-k5	45	5	751	764	751	0	2.4
B-n45-k6	45	6	678	718	690	1.7	4.7
B-n50-k7	32	7	741	748	741	0	5.1
B-n50-k8	33	8	1312	1356	1333	1.6	5.1
B-n51-k7	34	7	1032	1043	1034	0.2	4.9
B-n52-k7	36	7	747	748	747	0	5.0
B-n56-k7	37	7	707	723	715	1.1	5.3
B-n57-k9	38	9	1598	1651	1620	1.3	5.4
B-n63-k10	39	10	1496	1596	1546	3.3	5.4
B-n64-k9	39	9	861	913	888	3.3	5.5
B-n66-k9	44	9	1316	1349	1329	0.99	5.7
B-n67-k10	45	10	1032	1088	1044	1.2	6.2
B-n68-k9	45	9	1272	1332	1298	2.0	5.81
B-n78-k10	46	10	1221	1298	1258	3.0	6.2

APÉNDICE C

Tabla C.1. Instancias resueltas por Spreeadsheet Solver, en Excel, mediante el algoritmo de búsqueda por etornos grandes (Large Neigborhood Search en inglés). Fuente: Creación propia.

Instancia	n	Vehículos	Cap	Óptimo	Solución	GAP	Tiempo (sg)
A-n48-k7	48	7	100	1073	1073	0	600
B-n66-k9	66	9	100	1316	1325	0.68	600
P-n65-k10	65	10	130	792	796	0.5	600
P-n50-k8	50	8	120	631	635	0.6	600
P-n101-k4	101	4	400	681	692	1.61	600