UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA

MAESTRÍA EN OPTIMIZACIÓN Y CÓMPUTO APLICADO

Implementación de un algoritmo heurístico de Recocido Simulado para el Problema de Enrutamiento de Vehículos con Capacidades Homogéneas

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRÍA EN OPTIMIZACIÓN Y CÓMPUTO APLICADO

PRESENTA

DALIA VANESSA ARCE ORTEGA

DIRECTOR DE TESIS

FEDERICO ALONSO PECINA

CO-DIRECTOR

MARCO ANTONIO CRUZ CHÁVEZ

REVISORES:

DR. JOSÉ CRISPÍN ZAVALA DÍAZ

DRA. IRMA YAZMÍN HERNÁNDEZ BÁEZ

DR. MARCO ANTONIO CRUZ CHÁVEZ

DR. FEDERICO ALONSO PECINA

M.O.C.A. VÍCTOR HUGO PACHECO VALENCIA

CUERNAVACA, MORELOS MARZO,2023

Agradecimientos

A…

Dios en primer lugar por nunca abandonar mis pasos y siempre guiar mi camino, por darme lo que necesito y no lo que quiero en el momento justo.

Mis padres y hermanos por su esfuerzo, sacrificio, apoyo moral y por no perder la fe en que podría lograrlo.

Mi asesor de tesis el Dr. Federico Alonso Pecina, por su paciencia, enseñanzas, apoyo y tiempo dedicado al desarrollo de éste Proyecto.

Los integrantes de mi Comité: Dr. Marco Cruz Chávez, Dr. José Crispín Zavala Díaz, Dra. Irma Yazmín Hernández Báez, Dr. Cruz Rosales Martín Heriberto, Dr. Federico Alonso Pecina y Dr. Víctor Hugo Pacheco Valencia por todas sus recomendaciones, correcciones y revisiones dadas al proyecto.

"El fracaso comienza cuando cesa el esfuerzo"

Resumen

Dentro de muchas de las empresas de México y el mundo y más específicamente en el campo de la logística de las mismas, existen muchos problemas generados al momento de la repartición de productos.

La investigación que se realiza trabaja con el modelo de Enrutamiento de Vehículos con Capacidades Homogéneas (por sus siglas en inglés Capacited Vehicle Routing Problem CVRP), dicho problema es la representación de un problema de reparto de bienes, el cual, genera rutas de entrega para varios vehículos que cuentan con una capacidad idéntica, y deben repartir productos a un conjunto de clientes dispersos geográficamente, el propósito del problema es minimizar los costos de cada ruta, esto, minimizando la distancia recorrida cambiando el orden de visita de los clientes.

En el CVRP se cuenta con un número total de clientes, un conjunto de vehículos y una capacidad para los vehículos, los 3 parámetros no deben ser sobrepasados y en caso de los clientes no puede faltar ninguno por abastecer.

El presente problema fue resuelto aplicando la metaheurística del algoritmo: Recocido Simulado e incorporando 3 vecindarios diferentes para la mejora de la solución inicial generada de manera aleatoria.

Posteriormente el algoritmo fue probado con instancias de la literatura con el fin de medir su eficacia en calidad de la solución y tiempo de ejecución, dichas instancias van desde los 31 a 80 clientes, perteneciendo a dos grupos distintos de instancias.

Los resultados obtenidos, fueron muy satisfactorios y se logró igualar el resultado óptimo conocido en más del 60% de las instancias con las que fue probado el algoritmo. En comparación con otros métodos heurísticos implementados al problema, los resultados reportados en ésta tesis demostraron ser competitivos y muy buenos en cuestión de calidad de solución.

Abstract

Within many of the companies in Mexico and the world, and more specifically in the field of their logistics, there are many problems generated when distributing products.

The investigation that is carried out works with the model of Routing of Vehicles with Homogeneous Capacities (for its acronym in English: Capacitated Vehicle Routing Problem CVRP). This problem is the representation of a problem of distribution of goods, which generates delivery routes for several vehicles that have an identical capacity, and must deliver products to a set of geographically dispersed customers. The purpose of the CVRP is to minimize the costs of each route, thus, minimizing the distance traveled by changing the order of customer visits.

In the CVRP there is a total number of customers, a set of vehicles and a capacity for the vehicles, the 3 parameters must not be exceeded and in the case of customers, none can be missing to supply.

This problem was solved by applying the metaheuristics of the algorithm: Simulated Annealing and incorporating 3 different neighborhoods to improve the randomly generated initial solution.

Subsequently, the algorithm was tested with instances from the literature in order to measure its effectiveness in terms of solution quality and execution time.

Índice general

Capítulo 1. Introducción

1.1 Antecedentes.

Diariamente las empresas de México y el mundo enfrentan problemas relacionados con la optimización de sus recursos, entre éstas encontramos a las empresas que tienen el servicio de transporte de productos o están dedicadas completamente a la entrega de mercancía.

En los últimos años la demanda de entrega de productos a domicilio ha incrementado considerablemente debido a muchas causas, una de las más significativas ha sido el impacto causado por la pandemia de Covid 19, en la que la mayoría de los consumidores solicitan el servicio de entrega a domicilio.

Las paqueterías y empresas dedicadas a la logística se han convertido en un servicio muy solicitado y con esto se ha incrementado la necesidad de las mismas de tener una planeación de rutas eficiente para sus entregas, la mala administración de dichos recorridos puede representar una pérdida grande y constante de recursos y a su vez generar problemáticas que indirectamente provoquen pérdida de ingresos importantes.

Uno de los problemas más frecuentes en la toma de decisiones logísticas, es encontrar la manera de reducir los costos de transporte y mejorar el servicio al cliente encontrando los mejores caminos que debería seguir un vehículo para minimizar el tiempo o la distancia de los recorridos, éste problema es conocido como: "Problema de Enrutamiento de Vehículos".

Dicho problema representa una gran oportunidad de optimización de recursos y disminución de costos si se toma con la seriedad debida, la mala planeación de rutas de reparto ha generado desde el surgimiento de la logística un gran desperdicio de tiempo y bienes, generando así pérdidas inconmensurables a empresas que brindan el servicio de entrega de productos.

El problema de enrutamiento de vehículos surgió como una generalización del problema del Agente Viajero (TSP por sus siglas en inglés Travelling Sallesman Problem), éste problema consiste en un vendedor que debe visitar un número finito de lugares (lugares donde se encuentran sus clientes), debe visitar a cada uno solo una vez y regresar al lugar de partida. La solución a éste problema consiste en construir la ruta que minimice la distancia que tiene que recorrer el agente. Planteado de otro modo consiste en encontrar la ruta más corta que visita cada ciudad de una lista dada sólo una vez y regresa a la ciudad de origen. De éste problema surgió el problema llamado: Multi-agente viajero (m-TSP), en ésta variante del problema, contamos con el mismo planteamiento con la diferencia de que contamos con m agentes viajeros y cada cliente debe ser visitado solo una vez exactamente por un agente; cada vendedor comienza en el mismo lugar, denominado depósito y debe regresar al mismo al terminar de visitar a sus clientes.

Dicho esto, el Problema de Enrutamiento de Vehículos (VRP por sus siglas en inglés Vehicle Routing Problem) consiste en una empresa que debe repartir cierto producto entre sus clientes y desea encontrar la ruta (o conjunto de rutas) con menor costo, que partiendo de un almacén visiten a cada cliente y regresen al mismo. El VRP puede ser visto como el m-TSP con las características adicionales de que a cada cliente se le asocia una demanda y cada vehiculo cuenta con una capacidad.

El origen el VRP data desde el año 1959, cuando George Dantzig y John Ramser [Dantzig y Ramser, 1959] introdujeron la primera definición del problema, ellos describieron una aplicación real acerca de la entrega de gasolina a las estaciones de servicio y propusieron el primer modelo matemático. En

éste artículo ellos definen dicho problema como: «la determinación de la ruta óptima para una flota de vehículos que parten de uno o más depósitos (almacenes) para satisfacer la demanda de varios clientes dispersados geográficamente».

Cinco años después, en 1964, Clarke & Wright [Clarke, y Wright, 1964] desarrollaron el primer algoritmo que resultó efectivo para resolver VRP, conocido como el algoritmo de ahorros, éste algoritmo se convirtió en una de las técnicas más populares para resolver VRP a través de heurísticas, consiste (en el principio) de combinar una solución de 2 rutas diferentes para formular una nueva ruta donde se validan posteriormente los ahorros.

Es así como se da el comienzo de grandes investigaciones y trabajos en el área de ruteo de vehículos. A partir de ese momento el estudio de ruteo de vehículos ha crecido enormemente, tanto en la construcción de modelos que se acerquen más a la realidad como en la búsqueda de métodos de solución que sean cada vez más eficientes. El VRP es un famoso problema de optimización combinatoria y ha sido estudiado intensamente desde su surgimiento debido a las grandes aplicaciones en el campo de la logística.

El problema de enrutamiento de vehículos puede ser aplicado a muchas áreas de distribución algunos ejemplos son: Recolección de basura, entregas de gasolina a sus estaciones de servicio, distribución de productos especializados, rutas y establecimiento de paradas para uso del transporte público, abastecimiento de tiendas de abarrotes o centros de distribución (CeDis), transporte escolar, generación de rutas para servicios de mensajería y paqueterías, etc.

1.2 Planteamiento del problema

De manera general las empresas en México, no aplican un método o una técnica de planeación de rutas para la entrega de sus productos, esto ocasiona una falta de metodología que les permita visualizar dónde iniciar el reparto y dónde terminarlo sin dejar fuera a ningún cliente o evitar entregas irregulares.

En el campo de la logística, o aún más específico el campo de transporte, el problema de enrutamiento de vehículos constituye en muchísimas empresas uno de los temas más importantes de las mismas, debido a la gran pérdida de recursos que pueden ser ocasionados por la falta de una metodología adecuada de planificación de rutas de distribución.

El problema de enrutamiento de vehículos responde la pregunta: “¿Cuál es el conjunto óptimo de rutas para una flota de vehículos que debe satisfacer las demandas de un conjunto dado de clientes?”, es decir, el problema requiere la entrega de cierto producto, almacenado en un único local a los clientes geográficamente dispersos que poseen cierta demanda.

El CVRP se encarga del servicio de una compañía de entrega, un depósito y un conjunto dado de vehículos, los cuáles se mueven en una red de carretera dada, para entregar los productos a un conjunto de clientes. Así el problema determina un conjunto de rutas (una ruta para cada vehículo dado que inicia y termina en el depósito) tal que todas las demandas de los clientes son satisfechas y el costo de la transportación está minimizado. Esto puede ser minimizando el costo total de transporte basado en la distancia total recorrida con los vehículos utilizados o también minimizando el número de vehículos utilizados satisfacer a todos los clientes.

La red de carretera puede ser descrita utilizando un grafo donde los arcos son las carreteras y los vértices representan la localización de los clientes y el depósito.

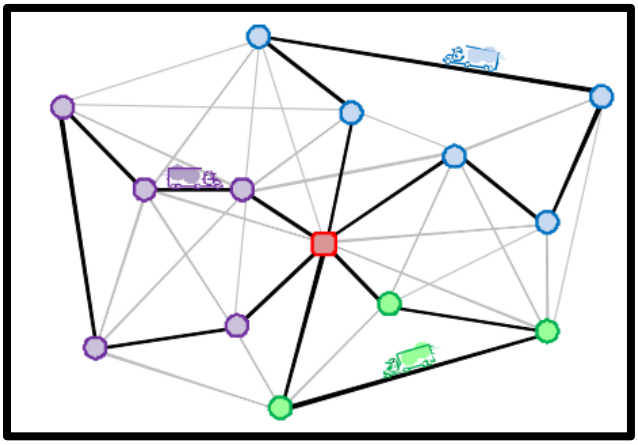


Figura 1 Grafo VRP

Debido a la dificultad y gran cantidad de tiempo requerido para la solución de grandes instancias con métodos exactos, muchos investigadores se han interesado en la resolución del VRP y sus variantes por medio de métodos distintos, con el fin de reducir el tiempo requerido para generar buenas soluciones.

La variante del VRP que se aborda en esta tesis es conocida como: "Problema de enrutamiento de vehículos con capacidades homogéneas" (CVRP por sus siglas en inglés Capacited Vehicle Routing Problem), en dicha variante se tiene un solo depósito y cada cliente tiene una demanda conocida que no se puede dividir. Cada vehículo tiene una capacidad fija y todas las capacidades de los vehículos son idénticas, así mismo cada cliente cuenta con coordenadas que nos ayudan a determinar su ubicación y también la distancia entre cada cliente y depósito.

El ruteo de vehículos permite establecer una estrategia para realizar la distribución adecuada de las mercancías, en los diferentes puntos en los cuales lo desee una organización, el propósito del problema es encontrar una ruta para cada vehículo, de modo que se minimicen los costos y no falte ningún cliente, cada ruta debe comenzar y terminar en el depósito.

Para tratar éste problema se utilizó Recocido Simulado, fue implementados de manera secuencial y probado en instancias conocidas de la literatura, utilizando varios benchmark con el propósito de evaluar y comparar el algoritmo desarrollado.

1.3 Hipótesis

Al implementar la metaheurística de Recocido Simulado, basada en un proceso de búsqueda por entornos para encontrar soluciones factibles y mejorar la calidad de estas con 3 vecindarios distintos en el problema de enrutamiento de vehículos con capacidades homogéneas se obtendrán al menos 50% de soluciones óptimas de un conjunto de instancias seleccionadas de la literatura

1.4 Justificación

El problema de ruteo de vehículos en las organizaciones pertenece al campo de transporte, distribución y la logística, dicho problema es de gran relevancia debido a las pérdidas económicas que puede representar la mala administración en las rutas, esto sucede por no contar con un buen mecanismo que permita generar un conjunto de rutas específicas que ayuden a minimizar el tiempo, costo y distancia en las entregas y por consecuencia minimizar las pérdidas económicas.

Algunos de los problemas que puede generar la mala administración de las rutas en la vida real son:

- Clientes faltantes: Al no planificar las paradas en cada recorrido puede llegar a suceder el olvido de alguna entrega, ignorando así la existencia de uno o más clientes que no recibirán su producto y esto generará pérdida de clientes y mala recomendación de éstos mismos a otros.
- Generación de ciclos: Cuando no se tiene un mecanismo organizado para indicar a los conductores que rutas seguir, puede generarse una gran cantidad de ciclos que sólo generarán pérdidas de recursos como combustible y tiempo.
- Accidentes: Cuando no hay consideración de distribución de clientes hacia rutas, es posible que a uno o más vehículos se les otorgue una carga excesiva de trabajo, esto puede llegar a generar un desgaste excesivo en el conductor y en los casos más extremos provocar accidentes automovilísticos provocados por el cansancio del conductor.
- Visitas dobles a clientes: No teniendo un mecanismo de planificación puede llegar a suceder que dos mismos vehículos visiten a un mismo cliente, esto generaría un incremento de desperdicio de recursos al hacerse una visita más que la necesaria a un cliente y también desperdiciando tiempo que podría ser empleado para otra visita.

El CVRP es un problema, que, al crecer el número de clientes que haya que satisfacer, crece también (y de manera exponencial) el número de combinaciones posibles de visitas a realizar, y con esto crece también el tiempo necesario para analizar y comparar dichas combinaciones; los problemas que cuentan con ésta característica en especial, han sido clasificados como problemas NP (Nondeterministic Polynomial Time), esto quiere decir que se necesita una cantidad de tiempo exponencial para encontrar la mejor solución a ellos y el CVRP es uno de ellos [Vigo y Toth, 2014].
La solución a problemas complejos, en particular los problemas NP-difícil o NP-HARD, implica un consumo de tiempo considerable durante su ejecución y también de memoria de almacenamiento, por ésta razón es de suma importancia desarrollar algoritmos que los resuelvan eficientemente.

El problema de enrutamiento de vehículos puede emplearse en distintas situaciones y ámbitos, satisfaciendo las necesidades de diferentes organizaciones que requieran seleccionar nodos en un orden específico para la entrega de productos y reducir las pérdidas económicas. Si no se cuenta con un sistema organizado dichas pérdidas pueden incrementarse excesivamente.

1.5 Objetivos

El objetivo general de la presente investigación consiste en implementar un algoritmo de Recocido Simulado que incorpore 3 vecindarios distintos para el problema de ruteo de vehículos con capacidades homogéneas con la finalidad de producir soluciones factibles y competitivas con las de la literatura. Para alcanzar el objetivo general se plantean algunos objetivos específicos que se mencionan a continuación:

- Programar un algoritmo que permita obtener soluciones factibles para el problema de enrutamiento de vehículos con capacidades homogéneas
- Incluir en el algoritmo al menos 3 vecindarios distintos para mejorar la solución inicial por medio de éstas y la metaheurística de Recocido Simulado.

- Evaluar el desempeño del algoritmo en la solución del problema de enrutamiento de vehículos con capacidades homogéneas en instancias de la literatura.
- Probar los resultados del algoritmo con instancias de la literatura con el propósito de alcanzar valores óptimos en algunas de ellas.

1.6 Organización de la tesis

El trabajo ha sido organizado de la manera siguiente:

- Capítulo 1: Se presentan los objetivos y justificación de la tesis
- Capítulo 2: Se plantean las técnicas heurísticas, su uso y se definen todos los aspectos del proceso de la metaheurística Recocido Simulado
- Capítulo 3: Se realiza la descripción del problema a tratar, se plantean y explican las variantes del problema y se realiza una revisión del estado del arte
- Capítulo 4: Se presenta la metodología de solución, la generación de solución inicial y se describen los vecindarios utilizados.
- Capítulo 5: Finalmente se muestran los resultados obtenidos con el presente algoritmo, también se presentan los Benchmark utilizados que proporcionan las instancias para las pruebas del algoritmo
- Capítulo 6: Se dan a conocer las conclusiones, trabajos futuros, y las referencias.

Capítulo 2. Optimización combinatoria

2.1 Técnicas de optimización

Las técnicas de optimización son herramientas que tienen como objetivo, desarrollar un proceso con el fin de determinar la mejor solución posible para un problema dado. Es decir, pretenden realizar la maximización de beneficios o la minimización de esfuerzos o pérdidas. El proceso de búsqueda de dicha solución es conocida como proceso de optimización.

El CVRP es un problema que pertenece al grupo de la optimización combinatoria u optimización discreta, la optimización combinatoria es una rama de la optimización en matemáticas aplicadas e informática, esta rama consiste en encontrar (dentro de un conjunto) un subconjunto que contenga las "mejores soluciones" a un problema.

Encontrar la mejor solución (solución óptima) en un conjunto finito de soluciones es un problema fácil que en teoría: solo se tiene que probar todas y cada una de las soluciones y compararlas entre ellas para ver cuál es la mejor. Sin embargo, el incremento exponencial de combinaciones de soluciones posibles (espacio de soluciones) de ciertos problemas matemáticos no permite obtener una solución en un tiempo "polinomial".

Por esto para resolver problemas de optimización combinatoria existen distintos métodos que pueden aplicarse y deben elegirse dependiendo de varios criterios. Un criterio que no debe ser tomado a la ligera, es precisamente el tamaño de la instancia que debe resolverse. Partiendo de esta característica nos encontramos con dos maneras básicas de abordar un problema: los métodos exactos y los métodos aproximados (heurísticas y metaheurísticas).

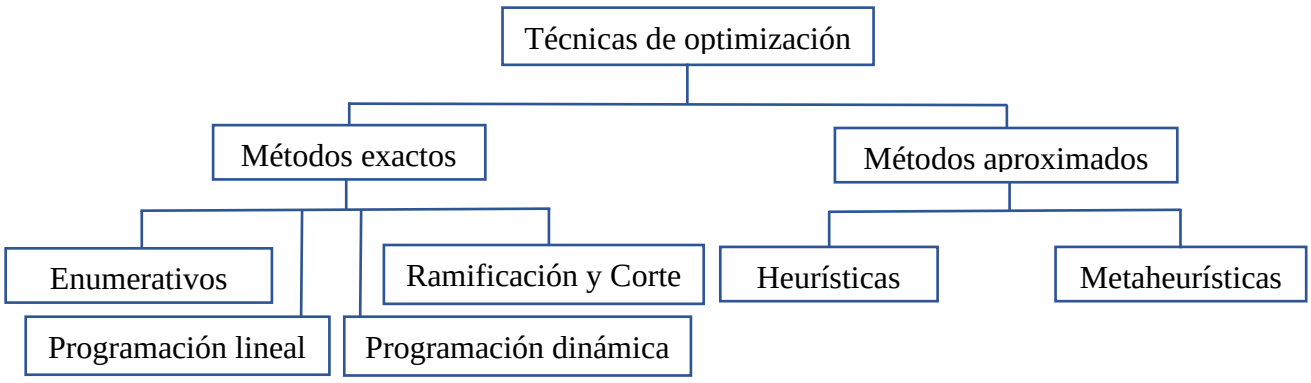


Figura 2 Técnicas de optimización

2.1.1 Métodos exactos

Los métodos exactos utilizan una manera ordenada de calcular o analizar todo el espacio de soluciones del problema, esto garantiza que la solución que se elige al finalizar la búsqueda es la mejor, es decir, es la solución óptima. Sin embargo, existen muchos problemas que no pueden ser resueltos con métodos exactos, esto debido a que crecen exponencialmente con el tamaño de la instancia, por esto, el tiempo requerido para utilizar un método exacto es extremadamente alto en algunos casos, creciendo de forma exponencial y haciendo imposible encontrar una solución óptima en un tiempo corto.

2.1.1.1 Métodos enumerativos

Dentro de los métodos exactos existen varias clasificaciones más, los métodos enumerativos y los métodos de búsqueda de Ramificación y corte, programación lineal y programación dinámica. Un método enumerativo busca secuencial y exhaustivamente recorriendo el espacio de soluciones en su totalidad, es decir, revisa todas y cada una de las opciones, combinaciones o permutaciones disponibles que puedan ser tomadas como solución factible para el problema y evalúa cada una de ellas para su posterior comparación.

2.1.1.2 Ramificación y Corte

El método de Ramificación y Poda, Ramificación y Corte o Ramificación y Acotación (del inglés Branch and Bound) fue desarrollado en 1960 para el problema general de Programación Lineal Entera (PLE) combinada o pura, dicho algoritmo interpreta el espacio de soluciones del problema como un árbol, donde cada rama representa una solución posterior a la actual, con esto el algoritmo se encarga de detectar la rama del árbol en la que las soluciones dadas ya no están siendo óptimas para "podar" o "cortar" dicha rama, evitando así continuar malgastando recursos y tiempo en los casos que se alejan de la solución óptima.

En el método de ramificación y poda se tienen múltiples opciones y muchos puntos dentro del algoritmo donde debe tomarse la decisión de elegir la siguiente variable de ramificación que dará paso al siguiente subproblema, esto, a veces es considerado una debilidad importante ya que, es muy difícil "prever" cual rama puede conducir a una solución mejorada.

De manera visual el algoritmo de Ramificación y Poda podría ser apreciado de la siguiente manera (Imagen tomada de [Taha H., 2012]) donde según el problema que se busca resolver, se toma la decisión de elegir la variable PL3 ubicada en el nodo 2 del árbol, como primera variable de ramificación y posteriormente la variable PL4 ubicada en el nodo 4 del árbol.

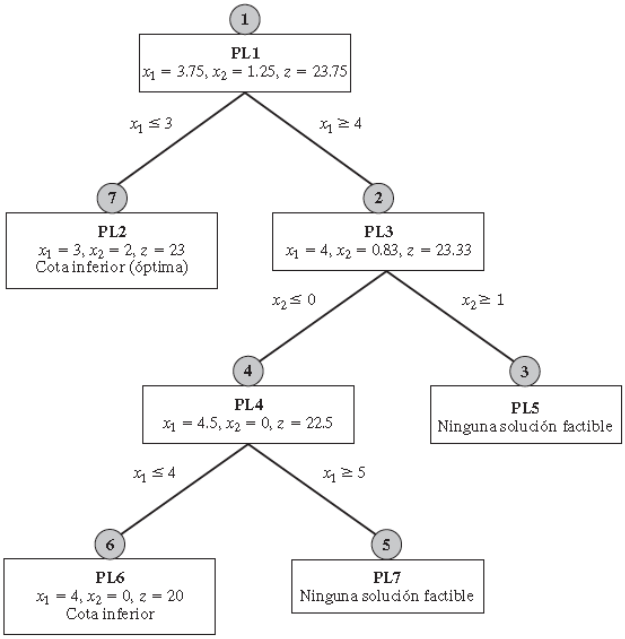


Figura 3. Árbol Ramificación y Poda

2.1.1.3 Programación Lineal

"Programación Lineal" (PL) es el nombre que se le da al cálculo de la mejor solución, a un problema modelado como un conjunto de relaciones lineales. El objetivo primordial de la Programación Lineal es optimizar, es decir, maximizar o minimizar funciones lineales, en varias variables lineales, con restricciones lineales (sistemas de inecuaciones lineales), optimizando una función objetivo también lineal.

En el caso del método de búsqueda de Programación Lineal el término "programación" se refiere a la disposición de un plan, en lugar de programar en un lenguaje de computadora.

Así pues, la Programación Lineal es un método mediante el cual se optimiza, ya sea maximizando o minimizando, una función objetivo. Esto, tomando en cuenta distintas restricciones dadas.

Los principales elementos de la programación lineal son los siguientes:

- Función objetivo: Es aquella función que se optimiza, ya sea maximizando o minimizando su resultado.
- Restricciones: Son aquellas condiciones que deben cumplirse al optimizar la función objetivo. Puede tratarse de ecuaciones o inecuaciones algebraicas.

2.1.1.4 Programación Dinámica

La programación dinámica es un método de optimización por etapas que divide el problema recursivamente en problemas más simples, se basa en una "subestructura óptima", significa que trata de usar soluciones óptimas de subproblemas para encontrar la solución óptima del problema en su conjunto. La Programación Dinámica se utiliza principalmente para reducir el tiempo de ejecución de un algoritmo, dado que dividir el problema en subproblemas suele volver más simple el procesamiento y esto en ocasiones resulta mejor que intentar resolver el problema en su totalidad.

Este método está inspirado en el principio de optimalidad de Bellman que dicta que: "dada una secuencia óptima de decisiones, toda subsecuencia de ella es, a su vez, óptima", dicho método intenta:

1. Dividir el problema en subproblemas más pequeños.
2. Resolver estos problemas de manera óptima usando este proceso de tres pasos recursivamente.
3. Usar estas soluciones óptimas para construir una solución óptima al problema original.

Existen dos tipos de programación dinámica: La Programación Dinámica Determinística, en la que se utilizan datos que se conocen con certeza y la Programación Dinámica Probabilística donde se usan datos que no se conocen con certeza pero que se determinan a través de distribuciones de probabilidad.

2.1.2 Heurísticas y metaheurísticas

Para el tipo de problemas que requieren demasiado tiempo de procesamiento y su complejidad crece con el tamaño del problema es que surgieron los métodos aproximados, algunos de ellos conocidos como heurísticas y otros como metaheurísticas, dichos métodos producen soluciones de alta calidad sin utilizar demasiado tiempo de procesamiento; crean soluciones que, aunque no garantizan ser la óptima son soluciones de calidad generadas en tiempo razonable.

No siempre es recomendable utilizar heurísticas o metaheurísticas para la resolución de problemas, estos métodos se utilizan cuando:

a) No existe ningún método exacto para la solución para el problema.
b) Existe un método exacto para resolver el problema, pero utilizarlo consume un costo computacional muy grande (requiere mucho tiempo para ofrecer una solución óptima).
c) Se requiere una solución rápida al problema o no se requiere la solución óptima para el mismo (limitación de tiempo)
d) Se utilizan como pasos intermedios para implementar una técnica posterior a su aplicación.

Una heurística es un método que produce buenas soluciones para un problema dado disminuyendo el tiempo de procesamiento pero a su vez sin asegurar encontrar la solución óptima, las heurísticas tienen algunas características que complican o incluso impiden utilizarlas de manera general: a) pueden quedar fácilmente atrapadas en "óptimos locales" y b) se diseñan e implementan para un problema en específico, es decir, están diseñadas para resolver un problema dado y no pueden usarse en otras condiciones: si hay modificaciones en el problema debe adaptarse la heurística con cada cambio.

Una metaheurística es una estrategia inteligente que mejora algoritmos heurísticos. Son una clase de algoritmos de aproximación que combinan el enfoque heurístico tradicional con estrategias de exploración del espacio de soluciones. Las metaheurísticas, por medio de estas estrategias de exploración (a menudo basadas en procesos biológicos, poblaciones, inspirados en la naturaleza, etc.) permiten salir de óptimos locales y continuar buscando soluciones aún mejores a éstas. Las metaheurísticas son algoritmos de propósito general y son descritas a menudo como metodologías de alto nivel.

La diferencia entre heurísticas y metaheurísticas radica principalmente en que las metaheurísticas tratan de escapar de óptimos locales realizando una orientación de búsqueda por medio de estrategias adicionales.

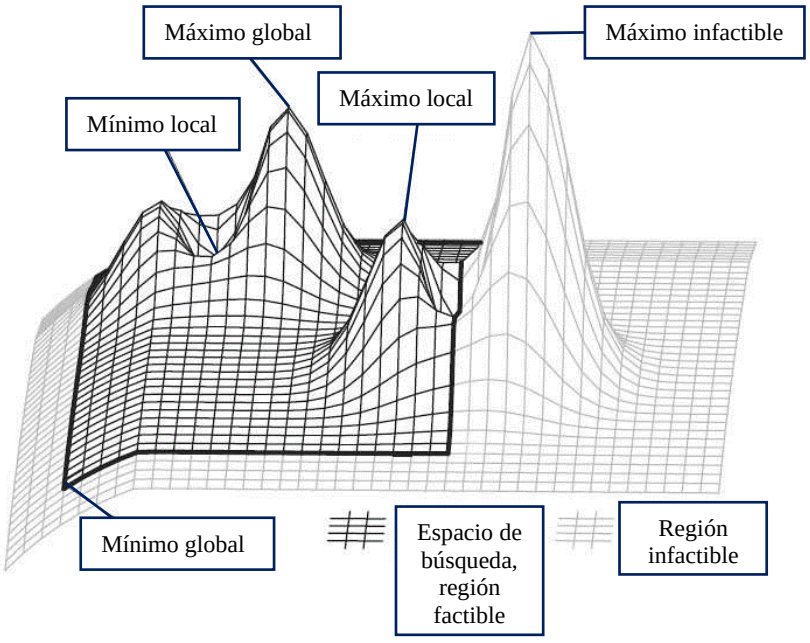


Figura 4. Mapa visual de Óptimos

Imagen tomada de [Granada y Santa, 2016].

Tanto heurísticas como metaheurísticas funcionan con un proceso que puede incluir:

- **Vecindarios:** la vecindad de una solución es un subconjunto de soluciones que pueden generarse mediante una modificación de la solución inicial (por permutación, por extensión, por mutación, por eyección, etc.).
- **Exploración:** la exploración consiste en recopilar datos de todo el vecindario completo. Se salta de una solución a otra calculando su función de costo.
- **Criterio de aceptación:** este criterio utiliza la información recolectada para definir las áreas de "interés" del área de investigación que forma el vecindario.
- **Memoria:** la memoria tiene en cuenta el aprendizaje del algoritmo y permite determinar las zonas susceptibles de tener un óptimo global. Si las nuevas soluciones o los criterios de parada ya no permiten mejorar la solución, el algoritmo se detiene. De lo contrario, regresa al paso 1.
  Ciertos algoritmos solo funcionan con criterios de detención, por lo que estamos hablando de metaheurísticas sin memoria.

Tabla 1. Técnicas de Optimización

Técnicas de optimización			
Métodos exactos	Heurísticas	Metaheurísticas	Hibridaciones
Son aquellos métodos que conducen a una solución óptima y llegan a una solución factible (entera). Los métodos exactos deberían utilizarse sólo para problemas de tamaño pequeño.	Una heurística es un algoritmo que permite obtener soluciones de buena calidad para un problema dado, esto permite tener menores tiempos de ejecución, pero sin asegurar la optimalidad de la solución	Las metaheurísticas son estrategias inteligentes basadas en analogías de la naturaleza para diseñar o mejorar algoritmos heurísticos.	En los métodos híbridos se combinan aspectos de varias heurísticas, metaheurísticas o algoritmos exactos con la finalidad de obtener lo mejor de ellos.

2.2 Complejidad

Aplicar una búsqueda exhaustiva para encontrar la solución de problemas de optimización combinatoria es un trabajo muy grande, tardado y robusto.

Enumerar todas las posibles soluciones para al final elegir la mejor de ellas es inviable en la mayoría de los casos porque el número de posibles soluciones crece exponencialmente con el número de variables del problema. Cuando se quiere resolver un problema de optimización combinatoria se debe saber qué tan difícil será encontrar una solución óptima. Una forma de medir este grado de dificultad es dada por la noción de *teoría de la complejidad*.

La teoría de la complejidad es parte de la teoría de la computación, dicha parte, estudia los recursos requeridos durante el cálculo para resolver un problema [Garey y Johnson, 2003]. Los recursos comúnmente estudiados son: el tiempo (número de pasos de ejecución de un algoritmo para resolver un problema) y el espacio (cantidad de memoria utilizada para resolver un problema). Con base en el estudio de estos recursos la teoría de la complejidad establece una clasificación de problemas, como problemas P (polinomial), NP (Nondeterministic polinomial), NP completo y NP duro [Parra y Chavez, 1998].

Asimismo, para conocer el grado de complejidad que puede llegar a tener un problema, se hace uso del modelo computacional conocido como "Máquina de Turing", con dicho modelo podemos obtener una clasificación de los problemas en base a su grado de complejidad.

Originalmente la máquina de Turing fue inventada en 1936 por el matemático inglés Alan Turing, definida como una "máquina automática", ésta no está diseñada como una tecnología de computación, sino como un dispositivo que representa una máquina de computación; en palabras de [Hopcroft et al., 2007] "la máquina de Turing se ha reconocido como un modelo preciso para representar lo que es capaz de hacer cualquier dispositivo físico de computación".

Así pues en base a éste modelo de la Máquina de Turing se han clasificado los problemas por su grado de complejidad para resolverlos, detectando que existe una línea de división muy importante, dicha línea divide a los problemas que se pueden resolver en tiempo polinómico y los que requieren un tiempo exponencial o mayor para ser resueltos. Los problemas que necesitan un tiempo polinómico casi siempre pueden solucionarse en un tiempo que podemos considerar como: tolerable, mientras que aquellos que precisan un tiempo exponencial, generalmente, no pueden resolverse (excepto para casos sencillos).

Así, es como se llega a los problemas intratables, clasificados como NP (Nondeterministic Polynomial Time), que, como el número de variables que componen el problema es extremadamente grande se piensa que son imposibles de resolver en un tiempo razonable. La teoría de la "intratabilidad" se refiere a las técnicas que demuestran que existen problemas que no pueden resolverse en un tiempo polinómico. Dentro de este grupo podemos encontrar problemas como: la calendarización de cursos universitarios, el problema del agente viajero, el problema de la mochila y más.

La clase *P* consta de todos aquellos lenguajes o problemas aceptados por alguna máquina de Turing que funciona en un tiempo polinómico, como una función de su longitud de entrada. La clase *NP* es la clase de lenguajes o problemas que son aceptados por una MT no determinista que opera en un tiempo polinómico limitado y lleva a resolver una secuencia de opciones no deterministas.

Como se mencionó anteriormente el problema de enrutamiento de vehículos es un problema considerado en lenguaje computacional como Np- duro o NP – Hard [Garey y Johnson, 1979]. Esto equivale a decir, que el tiempo que tarda el problema tratando de encontrar una solución óptima crece con el tamaño del problema. Entonces, se puede decir que entre más grande sea el problema más complicado será encontrar una solución óptima, o en su defecto una solución que sea factible.

Michael Gendreau afirma que los métodos exactos implementados en el CVRP son eficientes en problemas de hasta 50 clientes, debido a restricciones de tiempo computacional [Azi N., 2010].

Capítulo 3. Recocido Simulado

3.1 Sistema físico

De manera física recocer es un proceso o tratamiento térmico cuya finalidad es obtener estados de baja energía de un sólido mediante un baño térmico, dicho de otra manera, su finalidad es el ablandamiento, la recuperación de la estructura o la eliminación de tensiones internas principalmente en metales.

Figura 5. Recocido sistema físico

El proceso físico de recocido primero reblandece el sólido mediante su calentamiento a una temperatura elevada y luego lo va enfriando lentamente hasta que las partículas se van colocando por sí mismas en el estado fundamental del sólido, para dejar que se enfríe lento, habitualmente, se va apagando el horno dejando el metal en su interior para que su temperatura disminuya de forma progresiva. El proceso finaliza cuando el metal alcanza la temperatura ambiente.

El proceso de recocido es una técnica para modificar el estado de un material y alcanzar un estado óptimo mediante el control de la temperatura. Inicia con el calentamiento del material a una temperatura alta, para luego enfriarlo lentamente, manteniendo en cada etapa una temperatura por cierto tiempo.

El recocido se realiza en tres etapas: primero se calienta el material hasta la temperatura de recocido, después se mantiene la temperatura durante un tiempo determinado. Por último, se deja enfriar el material lentamente, si se disminuye la temperatura demasiado rápido, se pueden causar defectos en el material y el recocido no se logra, por eso es importante esperar un tiempo antes de volver a realizar otra disminución, para que con éste procedimiento se logre un estado estable en el sólido.

A medida que se disminuye la temperatura, el material va modificando su configuración (modifica la distribución de sus elementos), y si la temperatura desciende muy rápido las moléculas no podrán adoptar configuraciones cada vez más estables de energía. En esta técnica el enfriamiento del material causa una transformación de un estado de desorden a un estado de orden.

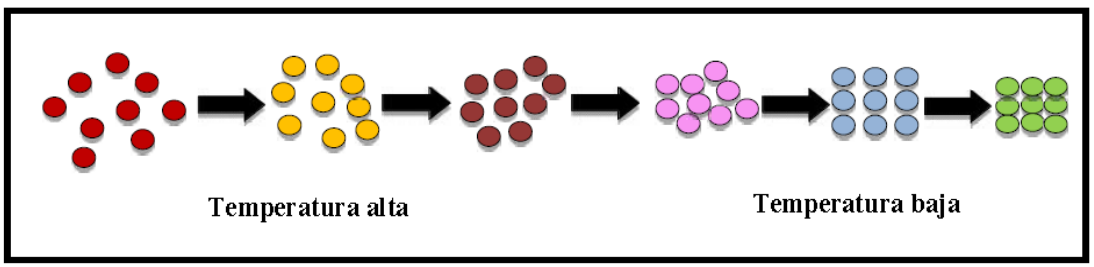


Figura 6. Distribución de elementos por temperatura

3.2 Metaheurística

La técnica metaheurística de Recocido Simulado (RS), Simulated Annealing (SA) (o enfriamiento simulado) es una analogía que simula el proceso físico de enfriamiento del recocido de sólidos, es un algoritmo de búsqueda por entornos para problemas de optimización combinatoria que está basado en el "principio de empeoramiento", esto quiere decir que, para escapar de óptimos locales, el algoritmo permite ciertos movimientos que empeoran la función objetivo.

El algoritmo es un procedimiento basado en "trayectorias", esto implica qué por medio de una solución inicial como punto de partida, va generando una trayectoria o camino (soluciones vecinas) en el espacio de soluciones que explora a través de perturbaciones (modificaciones) de la solución inicial.

Como se mencionó anteriormente el algoritmo RS es una analogía del proceso físico, una analogía corresponde a dos cuestiones diferentes, pero con una raíz común: es la relación de semejanza entre dos cosas que no son semejantes completamente.

A continuación, presentamos las analogías del sistema físico de recocido con el algoritmo RS basadas en [Siarry P., 2016].

Tabla 2. Analogías de Recocido Simulado

Sistema físico (termodinámica)	Problema de optimización
Estados del sistema	Soluciones factibles
Energía	Función de costo (función objetivo)
Temperatura	Parámetro de control T
Estado estable	Solución óptima

Basado en esto se pueden establecer las siguientes correspondencias:

- Las soluciones factibles generadas corresponden a los estados de transición del sistema físico, entonces, una alternativa vecina de un problema de optimización combinatoria (POC) corresponde a una pequeña distorsión al estado.
- El valor de la función objetivo en un POC es equivalente a la energía del estado actual del sistema.
- La temperatura del sistema físico se traslada al problema de optimización como parámetro de control, éste es el encargado de guiar el proceso del algoritmo generando una disminución de temperatura con cada iteración.
- Encontrar una buena solución en un POC es equivalente a encontrar estados de baja energía en el sistema real.

De esta manera se traslada el proceso físico de recocido de metales a la solución de un problema de optimización combinatoria, por medio de la temperatura se minimiza la función objetivo que es similar a la energía del material, así la temperatura se convierte en el parámetro de control del algoritmo y de igual manera que en el sistema físico esta temperatura debe conducir a un estado óptimo o solución óptima.

En cada iteración, la metaheurística de recocido simulado por medio de un estado actual S evalúa algunos vecinos y de manera probabilística decide entre efectuar una transición a un nuevo estado S' o

quedarse en el estado actual *S*. En el ejemplo de recocido de metales, el estado *S* se podría definir en función de la posición de todos los átomos del material en el momento actual; el desplazamiento de un átomo se consideraría como un estado vecino del estado actual.

El vecindario de un estado *S* está conformado por todos los estados a los que se puede llegar a partir de *S* mediante un cambio en la conformación del sistema (modificación de la solución inicial); los vecinos son generados mediante distintos métodos de perturbación de *S*.

El método de generación de los vecindarios es fundamental para encontrar una solución óptima global al problema. El método de Recocido Simulado basado en buscar siempre un estado vecino mejor (con energía más baja) que el actual, se detienen en el momento que encuentran un mínimo local de energía o cuando cumple el número de iteraciones establecido con anterioridad.

3.3 Descripción del funcionamiento del algoritmo

El algoritmo de Recocido Simulado está conformado por dos ciclos: uno externo (o ciclo de temperatura) que es controlado por el factor de equilibro térmico, y un ciclo interno (o ciclo de Metrópolis) que es regulado por un número total de iteraciones permitidas para buscar la mejor solución a cada temperatura. Entonces podemos decir que el método de Recocido Simulado está formado principalmente por el algoritmo de Metrópolis, éste lo conforma en su mayoría. Para iniciar con el algoritmo es necesario definir ciertos parámetros de control, cuyo valor puede cambiar dependiendo del problema a resolver, el modo de implementación del algoritmo y el objetivo de éste, ya que dichos parámetros siempre deben sintonizarse (modificarse y realizar pruebas) posteriormente con el objetivo de elegir los valores que generen mejores soluciones para el problema. Los parámetros a utilizar en el algoritmo de Recocido Simulado se describen a continuación:

Temperatura inicial: La temperatura inicial (como se ha mencionado anteriormente) debe ser un valor alto, esto con el fin de que exista un espacio que permita que muchos movimientos (o soluciones vecinas) sean aceptados. Para determinar este parámetro se puede inicialmente sólo seleccionar una temperatura que parezca alta y al ejecutar el algoritmo durante un tiempo corto, realizar pruebas mediante el porcentaje de aceptación con el fin de sintonizar el valor y elegir el que sea más conveniente. Otro método que ha sido implementado es utilizar el costo de la primera solución factible generada (solución inicial) como temperatura inicial, y posteriormente realizar la sintonización.

Factor de decremento (Alfa) o factor de enfriamiento: Se refiere a la velocidad de disminución de la temperatura a medida que avanzan las iteraciones del algoritmo. Existen diferentes maneras de realizar el decremento en la temperatura, una de las más utilizadas debido a que es simple y da buenos resultados es la ley de decremento geométrico que consiste en la multiplicación de un valor Alfa por la temperatura actual, con esto a cada iteración el valor de la temperatura disminuirá a la medida del valor que se asigne a Alfa. Normalmente el valor de alfa es elegido en un rango entre .85 a .99 [Kirkpatrick et al., 1983].

Número de iteraciones: El algoritmo, en cada nivel de temperatura realiza cierto número de iteraciones (repeticiones) del algoritmo de metrópolis, esto es equivalente en el sistema físico al tiempo que debe esperarse antes de disminuir la temperatura nuevamente, así para alcanzar un equilibrio es necesario un número de iteraciones constante (igual) en cada estado de energía, éste parámetro es también seleccionado con un criterio abierto para una posterior sintonización.

Temperatura final o criterio de parada: En el proceso físico la temperatura final debe ser la temperatura ambiente y de manera lógica en el algoritmo la temperatura debería reducirse a 0, sin embargo en la práctica la búsqueda llega a un estado de convergencia antes de alcanzar el valor de 0. Al igual que la temperatura inicial y los demás parámetros, el criterio de parada debe ser establecido al inicio y posteriormente sintonizado mediante pruebas ya que asignar un valor como 0 o aún más bajo puede incrementar excesivamente el tiempo de ejecución del algoritmo y/o volverse innecesario al obtener un estado de convergencia anterior al límite establecido.

El algoritmo de recocido simulado puede ser descrito como una serie de pasos que se describen a continuación:

1. Para comenzar, se genera una solución inicial factible que cumpla con todas las restricciones del problema, esta solución inicial puede ser generada de manera aleatoria o con algún otro método dependiendo del problema que se requiera resolver y las restricciones del mismo. Esta solución inicial servirá como punto de partida para generar posteriormente los vecindarios.
2. Se obtiene la función de costo de la solución inicial y se asignan los valores para los parámetros de control del algoritmo: la temperatura final (criterio de parada), se elige una temperatura inicial, un factor de decremento para la temperatura y se elige también el número de iteraciones para el ciclo interno, es decir, el número de ciclos de metrópolis que ejecutará el algoritmo.

   Algoritmo de metrópolis: Es el núcleo del recocido simulado y se basa en la técnica de Monte Carlo para generar una secuencia de estados del sólido en su proceso de enfriamiento, la generación de dichos estados se realiza mediante la perturbación de un estado actual. En el algoritmo de metrópolis, a partir de un estado de energía actual (solución actual) se genera un nuevo estado mediante una modificación en el sistema (generar un vecino) y se calculan los cambios de energía resultantes (diferencia de función de costo); si ésta modificación (vecino), origina una disminución en la energía (en el caso de tener una función objetivo de minimización), se acepta el cambio, y si ocurre un incremento de energía, el cambio se acepta asignando una probabilidad dada.
3. Teniendo ya los parámetros de control predefinidos se comienzan las iteraciones del algoritmo de metrópolis que podría ser descrito más detalladamente como:

   Ya contando con una solución inicial factible (que identificaremos ahora como **Sactual**) con cierto valor de energía o función de costo de la solución actual: **f(Sactual)**, se procede a generar aleatoriamente una nueva solución mediante una perturbación, dicha solución será una solución vecina (**Svecina**) y también se calculará su función de costo: **f(Svecina)**. Posterior a esto se realizará el cálculo de la diferencia de la función de costo entre ambas soluciones: ΔE = **f(Svecina) - f(Sactual)**.

   Si la diferencia de energía entre ambos estados es menor o igual a cero ($\Delta E \leq 0$), el estado nuevo (o solución vecina) es aceptado como el estado actual (solución actual). Si la diferencia de energía es mayor a cero ($\Delta E > 0$), el estado nuevo es aceptado con cierta probabilidad: P(ΔE), la cual está determinada por el criterio de Boltzmann.

Éste proceso se repite cierto número de veces: **nrep**, y cuando el ciclo iterativo se completa, se disminuye la temperatura y se comienza de nuevo con el ciclo interno (algoritmo de Metrópolis),

repitiendo así nuevamente la creación de soluciones nuevas, evaluación con su función de costo y criterios de aceptación de soluciones vecinas generadas, el algoritmo finaliza cuando la temperatura iguala el valor asignado a la temperatura final (criterio de parada) y al terminar indica cuál fue la mejor solución obtenida.

El planteamiento inicial del algoritmo fue concebido para problemas de minimización por lo que cuando $\Delta E < 0$ siempre será considerado que existe una mejora del estado actual, sin embargo, de manera general una mejora existe cuando se presenta un decremento en la energía del sistema. En optimización combinatoria, un movimiento (un vecino) se aceptará si su costo mejora, o en caso contrario, se aceptará si su probabilidad de aceptación es mayor que un número aleatorio uniformemente distribuido, esta estrategia permite que el algoritmo escape de óptimos locales.

En la medida en que evoluciona el proceso, la temperatura va disminuyendo y se incrementa la longitud de la cadena, por lo tanto, disminuye la probabilidad de aceptar soluciones de peor calidad. La codificación del problema define el espacio de búsqueda y por consiguiente la estructura de vecindad. Este aspecto juega un papel fundamental en la eficiencia del algoritmo debido al concepto de proximidad entre estados energéticos. En un sistema real un estado energético es una transformación continua del estado energético anterior y por lo tanto, una pequeña perturbación produce un pequeño cambio energético [Granada y Santa, 2016].

Longitud de la cadena de Markov en cada estado:
El ciclo interno del algoritmo de Recocido Simulado corresponde a una cadena de Markov, por lo que el número de iteraciones realizadas en ese ciclo corresponde con el valor de la longitud de la cadena de Markov.
La longitud de la cadena de Markov debe permitir al proceso alcanzar el cuasi equilibrio en cada nivel de temperatura. Más que un parámetro, la longitud de la cadena define cuándo se debe realizar un cambio de estado de energía. Una manera simple de seleccionar la longitud de la cadena es hacerlo de acuerdo con el tamaño del problema. Así, por ejemplo, un cambio de estado puede tomar lugar cuando una de las siguientes condiciones se cumple: **“**12*N* perturbaciones aceptadas” o “100*N* perturbaciones realizadas”, donde N es el número de variables del problema. Así, cuando se cumple una de las condiciones puede decirse que el sistema alcanzó el equilibrio termodinámico.
A medida que la temperatura disminuye, menos alternativas vecinas (perturbaciones) son aceptadas, y por lo tanto la exploración del vecindario en busca de una mejor alternativa se intensifica. Esto significa que a medida que la temperatura disminuye la longitud de la cadena de Markov aumenta implícitamente, con esto es posible establecer una relación entre la longitud de la cadena de Markov y la velocidad de enfriamiento: a temperaturas altas se requieren pocas iteraciones y a temperaturas bajas se requieren más.

A continuación, se presenta el pseudocódigo del algoritmo de Recocido Simulado descrito anteriormente.

Seleccionar una solución **Sactual** // **f(Sactual)** es su función de costo
Seleccionar un *criterio_de_parada* (**Tf**)
Seleccionar una temperatura inicial $T_0 > T_f > 0$
Seleccionar factor de decremento de la temperatura T_0
Seleccionar un número de iteraciones *nrep* // No. De ciclos de Metropolis
REPETIR
REPETIR
Generar aleatoriamente una nueva solución *Svecina* ϵ N(*Sactual*)
Obtener el valor de costo *f(s)* de *Svecina*
SI *f(Svecina)* < *f(Sactual)*
ENTONCES *S(actual)* ← *S(vecina)*
SINO
$\left(\exp - \frac{f(Svecina)-f(Sactual)}{T_0}\right) > random\ [0.1]$ } Criterio de Boltzman
ENTONCES *Sactual* ← *Svecina*
FIN SINO
HASTA QUE cuenta_iteraciones=*nrep*
T=a(t)

} Algoritmo de Metropolis

Figura 7. Pseudocódigo Recocido Simulado

Otros aspectos importantes que deben ser tomados en cuenta para la implementación del algoritmo de Recocido Simulado son los siguientes:

Generación de solución inicial: La generación de la solución inicial es una parte importante del algoritmo debido a que es el punto de partida que dará origen a la generación posterior de los vecindarios, la solución inicial puede generarse de manera aleatoria o con algún otro mecanismo predefinido. Dependiendo del problema que se desee resolver y su función objetivo, la solución inicial que se genera puede o no ser factible, aunque en la mayoría de los casos la solución inicial debe cumplir con el criterio de factibilidad, es decir, debe cumplir ciertas restricciones dadas por el problema, esto con el fin de mantener un orden desde el inicio de la ejecución del algoritmo y en algunos casos reducir posible tiempo posterior de procesamiento, tomando en cuenta desde el principio las restricciones del problema.

Función de costo: Es la función encargada de medir la calidad de la solución y su definición depende del problema que se requiere resolver. Lo que es importante resaltar es que ésta función debe calcular el costo total para cada nueva solución ya que será la medida para poder compararlas entre sí. Otro modo de no calcular todo el costo nuevamente (con el fin de ahorrar tiempo) es identificar la parte del cambio en la solución y recalcular sólo esa parte, sin necesidad de volver a calcular el costo de la solución completa, pero de igual manera depende del problema que se esté resolviendo el modo de la implementación del cálculo del costo.

Vecindarios: Definimos como vecindario al conjunto de soluciones vecinas generadas realizando configuraciones o modificaciones en cada iteración a partir de una solución inicial. La decisión de cómo generar la estructura de vecindad es muy importante debido a que es lo que da rumbo a la exploración del espacio de soluciones, la decisión depende en elegir cómo realizar las perturbaciones o movimientos para poder alcanzar una nueva solución a partir de otra.

Tomando en cuenta todos éstos aspectos que conforman al algoritmo de recocido simulado podemos proseguir con el planteamiento del problema, que es sobre el que se aplicó el algoritmo.

Capítulo 4. Problema de Enrutamiento de vehículos

4.1 Definición del Problema de Enrutamiento de Vehículos con Capacidades Homogéneas (CVRP)

El problema de enrutamiento de vehículos básico puede ser visto como un VRP o un CVRP, el problema consiste principalmente en obtener un conjunto de rutas usando la menor cantidad de vehículos y minimizando la distancia recorrida, dichas rutas deben comenzar todas en un único depósito y volver al mismo, ésto, después de haber satisfecho las demandas determinísticas (conocidas con anterioridad) e indivisibles de un conjunto de clientes que, dispersos geográficamente, tienen demandas distintas y donde la capacidad del vehículo es tomada como la restricción que da nombre al problema, es decir, la capacidad máxima de todos los vehículos debe ser la misma.

Se considera como "ruta" a un camino o recorrido ordenado generado previamente para un viaje, en éste caso para el propósito de entrega de productos a clientes, donde cada una de las rutas se asigna a un vehículo distinto y el número de clientes para cada ruta está establecido por la capacidad de los vehículos.

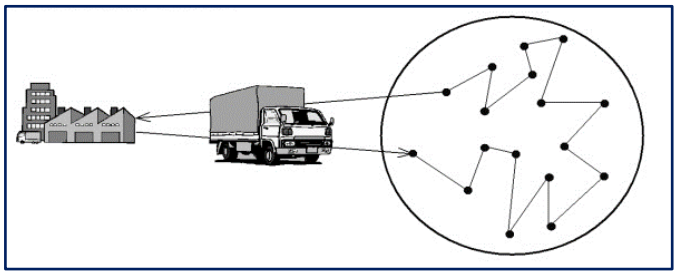


Figura 8. CVRP.

El problema de ruteo de vehículos es un problema considerado difícil de resolver considerado en lenguaje computacional como Np- duro o NP – Hard [Garey y Johnson, 1979]. Esto nos indica que, para poder encontrar una solución óptima, se necesita un esfuerzo computacional muy grande. Esto equivale a decir, que el tiempo que tarda el problema tratando de encontrar una solución óptima crece con el tamaño del problema.

El CVRP, se plantea en general como la distribución de una flota de vehículos que parten del depósito, o nodo inicial para cubrir con la demanda de los clientes, representados por nodos o vertices en un grafo. Los arcos que unen los nodos, pueden estar representados por distancias o costos de transporte.

Los componentes generales que se establecen para el estudio de un problema de ruteo son: los clientes, los vehículos, el depósito, las restricciones y los objetivos. A continuación, describiremos los elementos principales del CVRP.

- **Los Clientes:** Cada cliente tiene una ubicación dada y una demanda asignada, cierta demanda es determinística para éste problema y debe ser cubierta por algún vehículo. Cabe señalar que cada cliente representa un punto de parada para el vehículo y cada cliente debe ser visitado exactamente una vez.
- Una demanda determinística es aquella que es variable pero CONOCIDA, es decir, aunque varía de un cliente a otro es dada con anterioridad a la planeación de las rutas.
- **Los vehículos:** La flota de vehículos es homogénea, es decir que todos y cada uno de los vehículos poseen una capacidad idéntica, no se tiene un número límite de vehículos disponibles, pero es importante minimizar el número de vehículos a usar para la solución

debido al objetivo del problema, es decir: mientras más vehículos se utilicen el costo total incrementa.

- **El depósito:** En el CVRP o VRP básico se tiene un solo depósito que también es llamado en ocasiones "almacén" o "punto de partida", cada ruta debe comenzar y terminar en dicho punto y el mismo no puede ser un punto intermedio de ninguna ruta ya que contiene una ubicación, pero NO tiene asignada una demanda (porque no es un cliente).

Restricción del VRP

Las restricciones de un problema son un conjunto de limitaciones o reglas que se tienen para la generación de soluciones factibles, una solución factible es aquella que satisface todas las restricciones de un problema, dichas restricciones no pueden ignorarse ni pasarse por alto. Para el CVRP se cuentan con 3 restricciones:

- Visitar sólo una vez a cada cliente (nodo).
- Iniciar el recorrido y volver al punto de partida (depósito).
- No exceder la capacidad de los vehículos.

El modelo CVRP determina una cantidad de rutas con mínimo costo total, dicho costo es el resultado de la suma de los costos de las distancias obtenidas entre arcos pertenecientes al recorrido de la ruta.

Dicho ésto podemos hacer dos aclaraciones más:

- Todos los clientes (puntos de demanda) corresponden a entregas; y
- La distancia entre cada par de clientes puede ser igual en ambos sentidos (CVRP simétrico).

Por lo tanto, el CVRP se puede modelar como una programación entera mixta de la siguiente manera [Lin et al.,2009]: Red de distribución VRP (un depósito y 26 clientes).

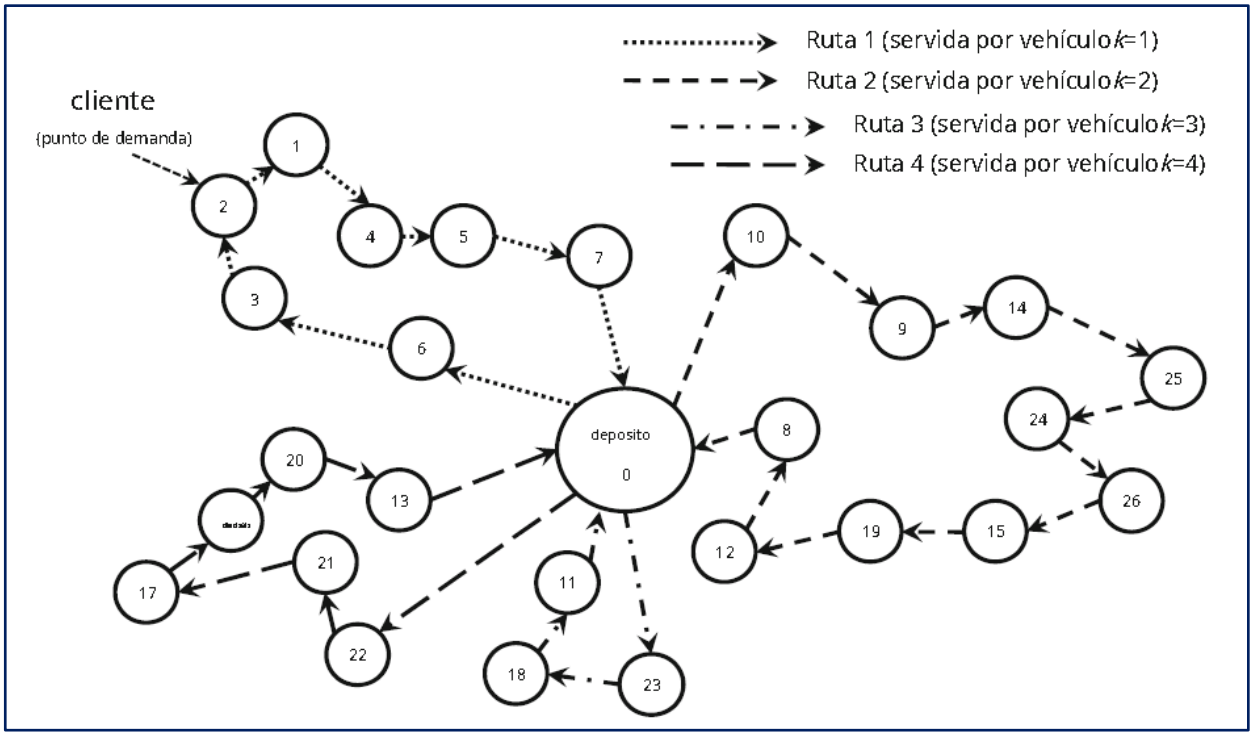


Figura 9. Red de distribución VRP.

4.2 Modelo matemático

Un modelo matemático es una representación simplificada a través de ecuaciones o funciones que utiliza fórmulas matemáticas para representar la relación entre distintas variables, parámetros y restricciones. Para ésta investigación se utilizaron los índices, parámetros y variables siguientes **[**Restrepo**,** 2008**]:**

INDICES
Los índices del modelo son:
i = nodo de partida *i* (1,2,…..,*n*)
j = nodo de llegada *j* (1,2,…., *n*)
n = nodos totales
k = Vehículo *k* (1,2,…., *K*)

PARÁMETROS
Los parámetros del problema son:
C_{ij}= Costo de transporte del nodo *i* al nodo *j*
d_j = Demanda en el nodo *j*
u =capacidad del recurso *k*
n = número de clientes

VARIABLES
Las variables que se definen son:
x_{ij}^{k}= 1 si se asigna el vehículo k para recorrer el arco del nodo i al nodo j.
cero(0) de lo contrario.
y_{ij}= 1 si se realiza el recorrido desde i hasta j o cero
(0) de lo contrario.

K= Número de vehículos a utilizar.

Lo cuál compone el modelo [González y González, 2006]:

Minimizar

$$\sum_{(i,j)\in A} C_{ij}\, Y_{ij} \qquad (1)$$

Sujeto a:

$$\sum_{i\le k\le K} x_{ij}^{k} = y_{ij};\ \forall\, i,j \qquad (2)$$

$$\sum_{i\le j\le n} y_{ij} = 1;\ \forall\, i \qquad (3)$$

$$\sum_{i\le i\le n} y_{ij} = 1;\ \forall\, j \qquad (4)$$

$$\sum_{i\le j\le n} y_{0j} = k; \qquad (5)$$

$$\sum_{i\le i\le n} y_{i0} = k; \qquad (6)$$

$$\sum_{i \le i \le n} \sum_{1 \le j \le n} d_i x_{ij}^k \le u;\ \forall\, k \qquad (7)$$

$$\sum_{i \in Q} \sum_{j \in Q} y_{ij} \le |Q| - 1 \qquad (8)$$

$$\forall \text{ subconjunto Q de } (1,2,\ldots n) \quad k \le K \qquad (9)$$

$$y_{ij} \in \{0,1\};\ \forall\, (i,j) \in A \qquad (10)$$

$$x_{ij}^k \in \{0,1\};\ \forall\, (i,j) \in A, \forall\, k \qquad (11)$$

El conjunto A se define como: A={(i,j);y_{ij}=1} (A cada par de arcos (i,j) tal que y=1).

De éste modo las ecuaciones del modelo se describen como:

(1) Es la función objetivo del modelo CVRP, minimiza la distancia recorrida.
(2) La ecuación dos se encarga de hacer explícita la relación entre las variables *x* e *y*. Si se usa en un vehículo la ruta de *i* a *j*, (algún $x_{ij}^k = 1$), entonces yij será uno. Se usa el vehículo k en el arco (i,j).
(3) En las ecuaciones (3) y (4) la variable y_{ij} nos indica la activacion del arco (indica que se hizo el recorrido). Ambas restricciones se aseguran de que todo cliente sea un nodo intermedio de alguna ruta (que no se olviden clientes) es, decir:
    (3) Para todo i (nodo de salida) existe un nodo de llegada (j).
    (4) Para todo j (nodo de llegada) existe un nodo de salida (i).
(5) Las ecuaciones (5) y (6) nos indican que k es la cantidad de vehículos utilizados en la solución y que todos los que parten del depósito o almacén deben regresar al mismo, esto quiere decir que se deben recorrer todas las rutas:
    (5) Todos los vehículos deben ir de 0 a j (depósito a clientes) o sea salir del almacén.
    (6) Todos los vehículos deben ir de i a 0 (clientes a depósito) o sea volver al almacén.
(7) Garantiza que no se sobrepase la capacidad de ningún vehículo (para todo k).
(8) La ecuación (8) vigila que la solución no contenga ciclos utilizando Q que representa el conjunto de nodos 1, 2, … n., en otras palabras, nos dice que la cantidad de rutas deben ser las rutas existentes).
(9) Limita el número de vehículos a utilizar en la solución.

(10) y (11) Nos indican que tanto la variable x_{ij}^k como la variable y_{ij}son binarias (solo pueden tomar el valor de 0 o 1).

4.3 Otras variantes del problema

Dentro del VRP básico existen dos determinaciones para el problema: El CVRP Simétrico y CVRP Asimétrico:

Tabla 3. CVRP simétrico y asimétrico

CVRP SIMÉTRICO	Se denomina CVRP Simétrico (Symmetric CVRP, SCVRP) cuando el costo de ir de un cliente i a un cliente j es el mismo que el costo de ir de un cliente j a un cliente i. Visualmente son representados por un grafo NO dirigido.

CVRP ASIMÉTRICO	En caso de que éstos costos no sean lo mismo se denominará al problema CVRP Asimétrico (Asymmetric CVRP, ACVRP). Un ejemplo de ésta situación son las calles de un solo sentido representadas visualmente por un grafo dirigido

También del CVRP básico se desprenden dos categorías más: CVRP Homogéneo y CVRP Heterogéneo:

Tabla 4. CVRP homogéneo y heterogéneo

CVRP HOMOGÉNEO	Se denomina CVRP Homogéneo al CVRP en el cual se aplican las mismas variables a cada ruta, cada vehículo o cada cliente. (Ejemplo: Las capacidades homogéneas, es decir, donde todas las distancias de los vehículos con iguales)
CVRP HETEROGÉNEO	En el CVRP Heterogéneo cada cliente tiene diferentes variables como son: más de un depósito, distancias asimétricas, entregas fraccionadas, etc. (Ejemplo: Capacidades heterogéneas, es decir, donde la distancia de cada vehículo puede ser diferente a la de los demás)

Sin embargo éstas características son sólo el inicio de una gran cantidad de variantes generadas por el VRP, dichas variantes serán analizadas a continuación.

De acuerdo a las diferentes necesidades que tiene una empresa a la hora de repartir sus productos, han ido surgiendo a lo largo del tiempo gran cantidad de características para el VRP que al irse agregando o quitando del problema van haciendo más complejo y a su vez más complicado de resolver de manera computacional.

Algunas de las variantes más utilizadas en investigaciones y a su vez más conocidas son:

- Problema del Agente Viajero (Travelling Salesman Problem TSP)
- Problema del multi Agente.viajero (Multi Travelling Salesman Problem m-TSP)
- Problema de Enrutamiento de Vehículos con Capacidades Homogéneas (CVRP)
- Problema de Enrutamiento de Vehículos con Capacidades Heterogéneas (HFCVRP)
- Problema de Enrutamiento de Vehículos con entrega y recolección (VRPB)
- Problema de Enrutamiento de Vehículos con ventanas de tiempo (VRPTW)
- Problema de Enrutamiento de Vehículos Multidepósito (MDVRP)
- Problema de Enrutamiento de Vehículos Periódico (PVRP)
- Problema de Enrutamiento de Vehículos Estocástico (SVRP)

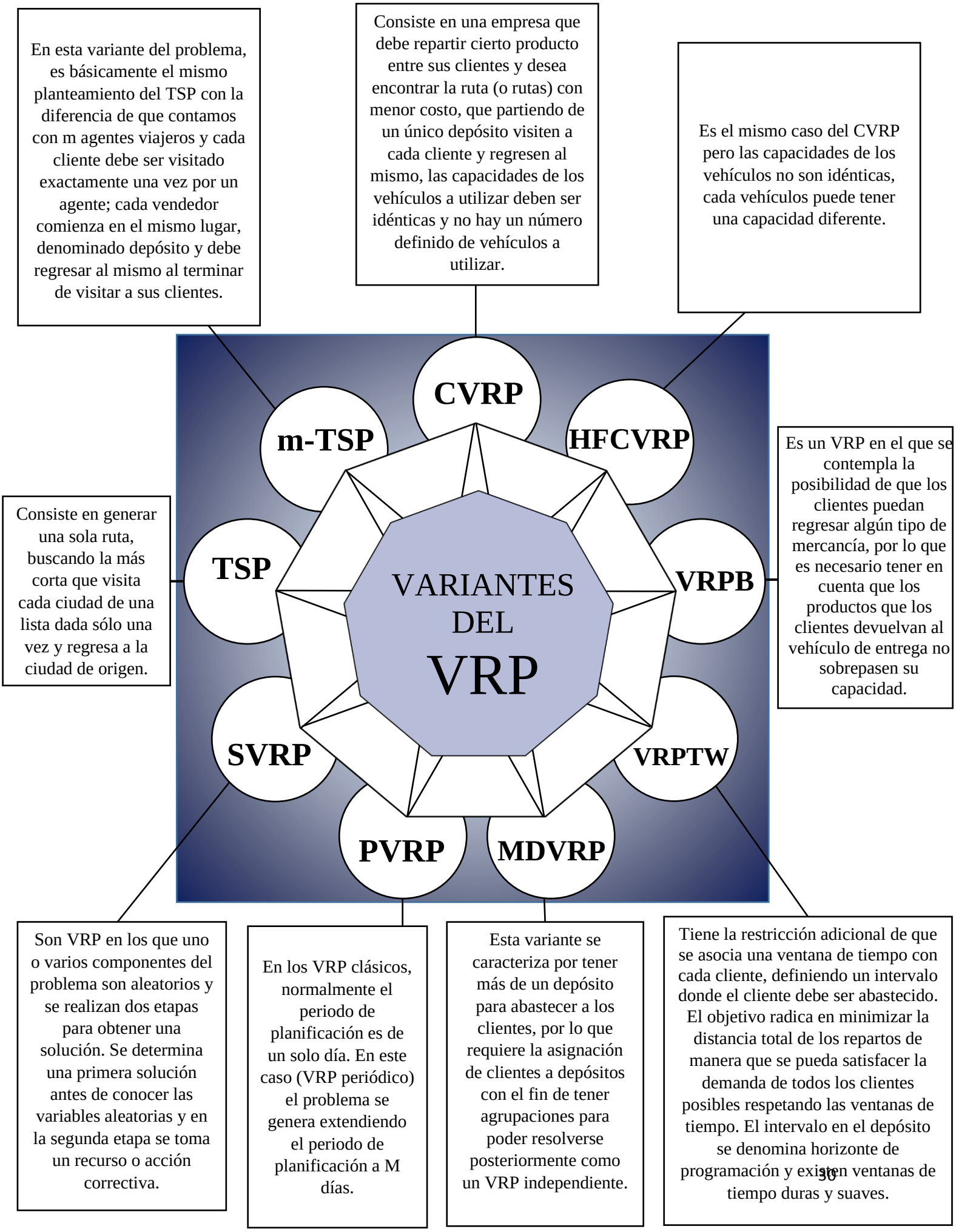


Figura 10. Variantes del VRP

4.4 Métodos aplicados para la solución del CVRP

El CVRP es un problema muy estudiado debido a su impacto y ha sido resuelto con distintos de los métodos, algunos de ellos se muestran a continuación:

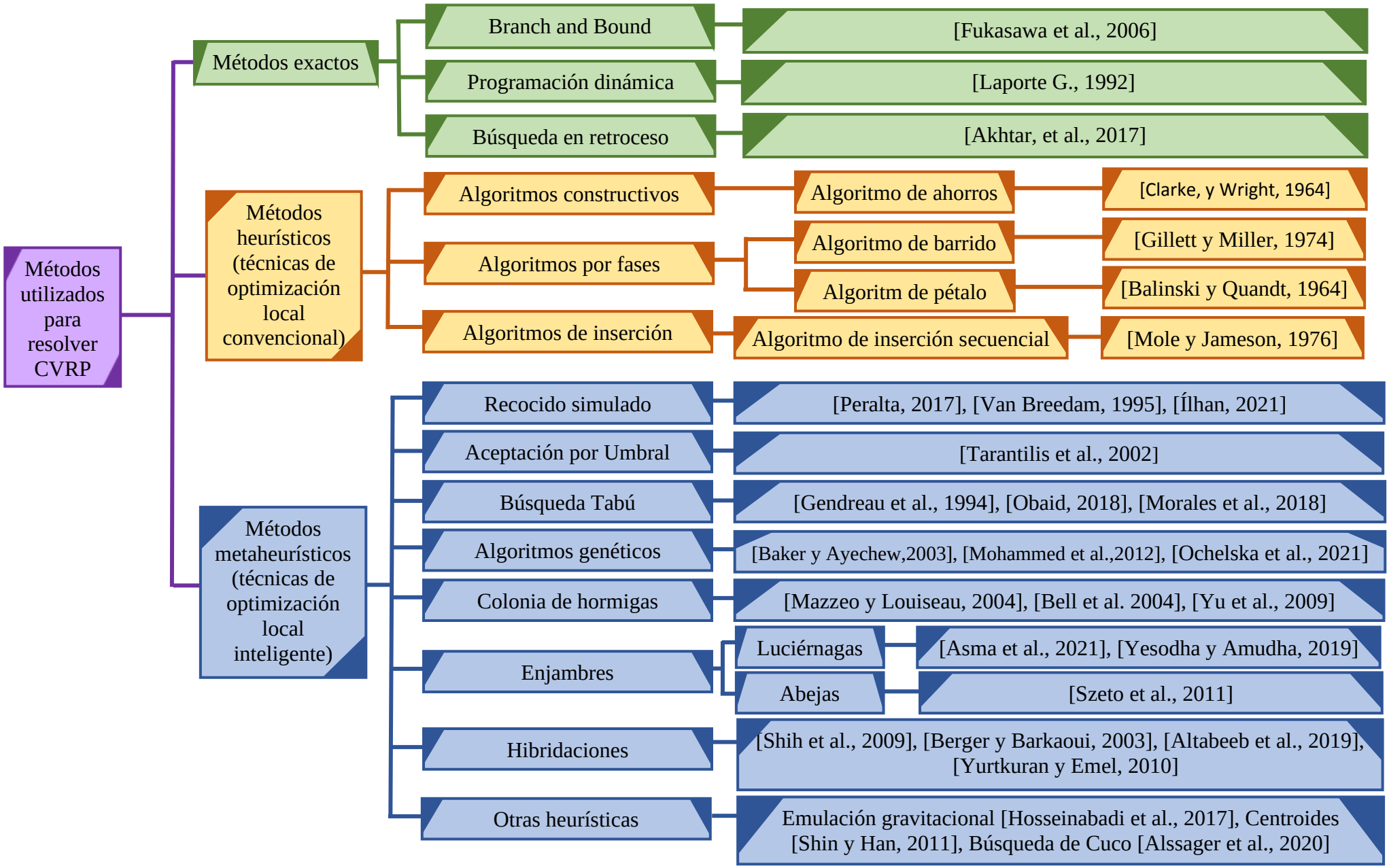


Figura 11. Métodos aplicados para la solución del CVRP

4.4.1Métodos exactos

4.4.1.1 Branch and bound

En [Fukasawa et al., 2006] se presenta un algoritmo que combina dos enfoques: trabaja sobre la intersección de dos politopos, uno asociado con una relajación tradicional sobre rutas y el otro definido por restricciones de límite, grado y capacidad.

En éste algoritmo cada cliente se representa como una caja que contiene su demanda, la idea principal del trabajo es combinar el enfoque de ramificación y corte con el enfoque de q-rutas (que interpretamos como generación de columnas en lugar de la relajación original de Lagrange) para derivar límites inferiores superiores.

La idea de combinar la generación de columnas y cortes para mejorar los límites inferiores es una de las novedades del trabajo (rara vez se había utilizado), ya que las nuevas variables duales correspondientes a cortes separados pueden tener el efecto indeseable de cambiar la estructura del subproblema de precios, haciéndolo intratable.

4.4.1.2 Programación dinámica

En [Laporte G., 1992] se menciona que los algoritmos exactos para el VRP se pueden clasificar en tres amplias categorías: 1) métodos de búsqueda directa de árboles, 2) programación dinámica y 3) programación lineal entera. Así pues como el número de algoritmos propuestos es muy grande, proporciona solo seis ejemplos representativos: dos métodos de búsqueda de árbol directo basados en diferentes relajaciones, una formulación de programación dinámica y tres algoritmos de programación lineal entera.

Para la programación dinámica plantea que es probable que el número de cálculos requerido sea excesivo en la mayoría de los problemas y el uso eficiente de la programación dinámica requiere una reducción sustancial del número de estados mediante un procedimiento de relajación, o utilizando criterios de factibilidad o dominancia.

4.4.1.4 Búsqueda en retroceso

[Akhtar et al., 2017] presenta un algoritmo de búsqueda de retroceso (backtracking) en un problema de generación de rutas de vehículos con capacidades homogéneas (CVRP) para optimizar las rutas de recolección de residuos para reducir los costos económicos y los impactos ambientales. El proceso típico de recolección de residuos involucra vehículos que parten del depósito: viajan en rutas fijas para recoger desperdicio visitando todos los lugares, mientras recolectan desechos, los vehículos mantienen sus motores en funcionamiento, incluso durante la carga y descarga de contenedores de basura.

El BSA (Backtracking search algorithm) fundamental se modifica aplicando una serie de algoritmos de mejora local y utiliza datos de contenedores inteligentes para recolectar desechos de manera eficiente. Los contenedores inteligentes pueden obtener datos de residuos en tiempo real a través de varios sensores, posteriormente se comienza con la implementación del BSA.

4.4.2 Métodos heurísticos

4.4.2.1 Algoritmo de ahorros

El algoritmo de ahorros presentado en [Clarke y Wright, 1964] es un algoritmo constructivo que en un principio consiste en combinar una solucion de dos rutas distintas para formar una ruta nueva donde por medio de una fórmula se valida el ahorro de la combinacion de soluciones, con ésta combinacion de rutas se van eliminando aristas. Se ve al problema como un grafo y con cada arista eliminada se genera un ahorro de distancia en las rutas.

4.4.2.2 Algoritmo de barrido

[Gillett y Miller, 1974] nos muestra un algoritmo denominado "de barrido" para resolver problemas de despacho de vehículos de mediana y gran escala con restricciones de carga y distancia para cada vehículo. Las ubicaciones que se utilizan para crear cada ruta se determinan de acuerdo con el ángulo de coordenadas polares de cada ubicación. Luego se utiliza un procedimiento iterativo para mejorar la distancia total recorrida en todas las rutas. El algoritmo tiene la característica de que la cantidad de cómputo requerida aumenta linealmente con el número de ubicaciones si el número promedio de ubicaciones para cada ruta permanece relativamente constante.

El algoritmo de barrido consta de dos partes, un barrido hacia adelante y un barrido hacia atrás. En la parte de hacia adelante, las ubicaciones se dividen en rutas que comienzan con la ubicación que tiene el ángulo más pequeño, tomando en cuenta que las ubicaciones se volvieron a numerar de acuerdo con el tamaño de su ángulo de coordenadas polares y el depósito es la ubicación 1.

Un segundo algoritmo, llamado algoritmo de barrido hacia atrás, es exactamente igual que el algoritmo de barrido hacia adelante excepto que forma las rutas en orden inverso. En la mayoría de los resultados presentados por el artículo, los dos procedimientos producen rutas diferentes y, en consecuencia, en algunos casos, distancias totales mínimas diferentes

4.4.2.3 Algoritmo de pétalo

En [Balinski y Quandt, 1964] se implementa un algoritmo de pétalo para resolver un problema real de entrega de camiones, en éste, se calculó un conjunto de diez problemas que surgieron de una situación comercial real utilizando un algoritmo conocido como: "plano de corte".

En éste algoritmo probado en 10 instancias reales, la solución entera está en alguna cara de baja dimensión (llamado pétalo) del conjunto poliédrico convexo "S" definido por las restricciones. Por lo tanto, las soluciones óptimas de programación lineal a los problemas son también las soluciones enteras óptimas, es decir, estas soluciones óptimas son puntos extremos de S.

4.4.2.4 Algoritmo de inserción secuencial

En [Mole y Jameson, 1976] se describe un algortimo secuencial de construcción de rutas que emplea un criterio de ahorro generalizado, el método para resolver el CVRP está basado en la generación secuencial de rutas que al principio considera la posibilidad de que cada vehiculo realice varios viajes, en consecuencia de ésto la utilización del vehículo es elevada, posteriormente se generaliza el criterio del algoritmo de ahorro de Clarke y Wright y con ésto cada trayecto queda libre de intersecciones.

De manera general el algoritmo descrito, se puede considerar una secuencia repetitiva de tres pasos: en el primer paso se determina la ubicación más ventajosa de cada cliente, después el segundo paso utiliza un criterio para identificar al próximo cliente que se colocará en una ruta emergente y en el

paso tres, se explora la posible resecuenciación de clientes en la ruta emergente utilizando métodos 2-opt.

4.4.3 Métodos metaheurísticos

4.4.3.1 Recocido Simulado

[Peralta, 2017] propuso dos algoritmos para tartar el CVRP, primero un algoritmo de recocido simulado secuencial apoyado en una estructura híbrida de vecindad y con reinicio como mecanismo de exploración y explotación del espacio de soluciones. El otro algoritmo es el de recocido simulado distribuido también con reinicio y una estructura híbrida de vecindad, el cuál fue desarrollado con la libreria de "paso de mensajes (MPI).

Ambos algoritmos desarrollados en éste trabajo se ejecutaron de manera secuencial y distribuida en un cluster.

En [Van Breedam, 1995] se informa sobre el uso de métodos de mejora basados en recocido simulado para el problema de generación de rutas para vehículos. Los métodos de mejora considerados tienen como objetivo la reubicación y/o intercambio de paradas o cadenas de paradas entre diferentes rutas, partiendo de una solución inicial factible. Los métodos de mejora basados en recocido simulado se comparan con sus alternativas de descenso, así como con otras implementaciones metaheurísticas en un conjunto de problemas de prueba clásicos y se informan los resultados.

El estudio también menciona que: se supone que los métodos de mejora, mejoran las soluciones viables a través de un mecanismo de búsqueda. Este mecanismo de búsqueda puede seguir un patrón recto como en la heurística de descenso o escalada. En consecuencia, esto conduce esencialmente a mejoras de la función objetivo hacia un óptimo local. Además, la calidad de la solución final depende en gran medida de la calidad de la solución inicial. Si, por el contrario, la regla de búsqueda no es sistemática y si se acepta un deterioro temporal del valor de la función objetivo, es posible contrarrestar las limitaciones de convergencia a óptimas locales antes mencionadas y la influencia de la solución inicial, propia de métodos de mejora local.

En el estudio de [Ílhan, 2021] se propuso un algoritmo de recocido simulado con operador cruzado denominado ISA-CO, los operadores de cruce se aplicaron en las soluciones en la población para acelerar la convergencia.

En el artículo se utilizó un algoritmo de recocido simulado basado en población. Las soluciones en la población se desarrollaron con operadores de búsqueda local de intercambio, codificación, inserción y reversión y el algoritmo mejorado de 2 opciones se utilizó para desarrollar las rutas que componen la solución. Los operadores de cruce parcialmente mapeado (PMX) y de cruce de orden (OX) se aplicaron a las soluciones en la población para acelerar la convergencia.

4.4.3.2 Aceptación por Umbral

[Tarantilis et al., 2002] describe un algoritmo de aceptación por umbral basado en listas (LBTA), su principal diferencia con respecto a un algoritmo de aceptación de umbral típico es que los valores del umbral utilizados en el criterio de aceptación de movimientos están determinados por una lista que se rejuvenece y adapta de acuerdo con la topología del espacio de solución del problema. Esto es: los algoritmos de aceptación por umbral típicos buscan iterativamente en el espacio de soluciones

guiados por un parámetro de control, éste parámetro se reduce a lo largo del algoritmo y se denomina umbral, cuando se generan soluciones nuevas por medio del mecanismo de perturbación del algoritmo éstas son evaluadas con las restricciones y el criterio de aceptación del movimiento (la diferencia de costo entre la solución nueva y la anterior debe ser inferior al umbral).

Si la nueva configuración generada cumple con los criterios mencionados anteriormente, reemplaza la configuración anterior y da lugar a una nueva. La innovación del algoritmo LBTA se basa en el hecho de que los valores de umbral utilizados en la implementación del criterio de aceptación de movimiento están determinados por una lista que se rejuvenece y adapta de acuerdo con el topología del espacio de solución del problema, con esto el algoritmo acepta soluciones vecinas de una solución actual que aumentan los costos más fácilmente, facilitando el escape de los puntos mínimos locales, aumentando así la posibilidad de encontrar una solución mejor.

4.4.3.3 Búsqueda Tabú

[Gendreau et al., 1994] presenta una heurística de búsqueda tabú para el problema de generación de rutas de vehículos con restricciones de capacidad y longitud de ruta. El algoritmo considera una secuencia de soluciones adyacentes obtenidas al eliminar repetidamente un vértice de su ruta actual y reinsertarlo en otra ruta.

Antes de comenzar las iteraciones y como primer paso se establece un contador de iteraciones. Al principio ningún movimiento es tabú, después, se realiza una selección de vértices aleatorios de una solución y se evalúan todos los movimientos posibles para todos los vértices seleccionados, posteriormente se identifica el mejor movimiento (es aquel que produce el menor valor de costo de solución). El movimiento identificado no es necesariamente implementado. Si el movimiento no se ha utilizado y el vértice se ha movido de la ruta R a la ruta Rs, se reinserta el vértice en R y se declara tabu hasta la iteración deseada. Posteriormente se realizan ajustes de penalización y para finalizar se detendrá en el número de iteraciones declarado en un inicio.

En [Obaid, 2018] la estructura del algoritmo está planificada con el objetivo de que el programa no requiera una base de datos sustancial para almacenar los datos, lo que acelera el uso de la ejecución del programa para adquirir la solución.

La idea fundamental del algoritmo de búsqueda tabú es evitar redundancias y rechazar la redundancia en la etapa siguiente. La búsqueda tabú continúa con la suposición de que no hay razón para tolerar otras (malas) soluciones a menos que se desvíe de una ruta ya explorada. Esto garantiza que se investigarán nuevas áreas de espacio para soluciones de problemas con el objetivo de evitar mínimos locales y finalmente encontrar una mejor solución. Para esta situación, la búsqueda tabú alienta a usar la utilización de la memoria para evitar un bucle sin fin, entonces, comienza con el progreso hasta los mínimos locales. Para abstenerse de recordar los medios utilizados, la estrategia registra los movimientos posteriores en al menos una lista tabú. El primer plan de la lista no era evitar que se volviera a repetir un movimiento pasado, sino salvaguardar que no se volviera a la misma solución tomando el mismo camino (es posible visitar a la misma solución, pero yendo en otra dirección distinta).

La metaheurística propuesta en [Morales et al., 2018] integra un proceso de solución en dos etapas: primero, se obtiene un conjunto de rutas CVRP factibles mediante un algoritmo de búsqueda tabú (Tabu-Search-TS), y segundo, se integra un Algoritmo Genético (Genetic Algorithm-GA) para

mejorar la factibilidad de cada ruta. El resultado de esto es una metaheurística de búsqueda tabú evolutive (Evolutive Tabu-Search-E-TS).

La metaheurística E-TS propuesta comienza con un ciclo hamiltoniano generado aleatoriamente que comienza y termina en la ubicación del depósito, éste ciclo representa una posible solución para el problema del agente viajero, ésta solución posteriormente se divide en subrutas para cumplir con las características del problema, generando así una solución inicial factible para CVRP, después sobre ésta solución se aplica el algoritmo de búsqueda tabú, mientras que el elemento TS de la metaheurística construye y mejora una solución CVRP, la metaheurística GA realiza una mejora adicional basada en operadores evolutivos. Esta mejora se realiza en cada ruta del CVRP y no altera el número de nodos o puntos de demanda atendidos por cada ruta. Por lo tanto, el GA no altera la asignación de puntos en cada ruta definida por el algoritmo TS.

4.4.3.4 Algoritmos genéticos

El artículo [Baker y Ayechew, 2003] realiza un estudio sobre la aplicación de un algoritmo genético al problema básico de enrutamiento de vehículos, para el que se describe: dados n clientes y m vehículos, el cromosoma para una solución individual tiene la forma de una cadena de longitud n. La solución de un problema de vendedor ambulante es requerida para cada vehículo con el fin de hacer la transición de una solución implícita a una explícita, permitiendo asociar un valor con cada miembro de la población.

Se toman dos pasos para mantener la mayor estructura posible. En primer lugar, se clasifica a los clientes y se enumeran para que los clientes consecutivos puedan ser atendidos por el mismo vehículo. Para problemas en los que los clientes se distribuyen aleatoriamente por el depósito, se clasifican de acuerdo al orden creciente del ángulo polar. Cuando los clientes están ubicados en grupos en lugar de aleatoriamente, se clasifican de acuerdo con una solución vecina más cercana al TSP, comenzando desde el depósito y visitando a todos los clientes. En segundo lugar, se intenta numerar los vehículos para que cualquier vehículo opere en aproximadamente la misma región para todos los miembros de la población. Luego, se aplican los procesos reproductivos habituales de GA a la población, generando nuevas soluciones que comparten ciertas estructuras de ruta con sus padres.

En el algoritmo genético que implementa [Mohammed et al., 2012] se realiza una representación cromosómica: el gen se codifica como números que se asignan al cromosoma. El número representa la secuencia de paradas (ubicaciones) en las que se entregan las mercancías. Esta codificación se usa para ordenar problemas en los que se corrigen algunos cruces y mutaciones para mantener la consistencia del cromosoma. Se tiene también una función fitness: el menor costo de cada ruta en este problema depende de la distancia, que es la única variable. Dado que es deseable acortar la distancia de la ruta, entonces la aptitud del cromosoma puede calcularse simplemente obteniendo la suma de las distancias para cada ruta.

[Ochelska et al., 2021] presenta la idea de implementar diferentes operadores genéticos, modificados para su uso con el CVRP, el núcleo del algoritmo genético descrito es un módulo completamente separado que está escrito de tal manera que puede usarse no solo para resolver problemas VRP, sino también cualquier problema factible para soluciones GA.

Cuenta también con algunos componentes modulares, uno de ellos es la representación cromosómica: los objetos que representan cromosomas son de una clase que implementa todos los operadores de

mutación para ellos. Tal enfoque proporciona la encapsulación de toda la lógica relacionada con el genoma y las operaciones realizadas en él. El último módulo requerido es un mecanismo de cálculo de función objetivo

4.4.3.5 Colonia de hormigas

[Mazzeo y Louiseau, 2004] Desarrollaron un algoritmo Ant Colony (ACO) para el CVRP basado en la técnica metaheurística introducida por Colorni, Dorigo y Maniezzo [Colorni et al., 1991] quienes se inspiraron en una analogía con el comportamiento de una colonia de hormigas en la vida real cuando busca comida y propusieron una técnica metaheurística: las hormigas son procedimientos que construyen soluciones a un problema de optimización.
Un tema importante del algoritmo desarrollado es el paralelismo: se construyen varias soluciones al mismo tiempo e intercambian información durante el procedimiento y utilizan información de iteraciones anteriores.

Otro autor [Bell et al. 2004] también trabajó con colonia de hormigas, en donde se inspiraron en el mecanismo con que las hormigas se comunican, esto de manera física se realiza mediante una sustancia química llamada feromona. Cuando una hormiga viaja, deposita una cantidad constante de feromona que pueden seguir otras hormigas. Cada hormiga se mueve de una manera algo aleatoria, pero cuando una hormiga se encuentra con un rastro de feromonas, debe decidir si seguirlo o no. Si se sigue el rastro, la propia feromona de la hormiga refuerza el rastro existente, y el aumento de feromonas aumenta la probabilidad de que la próxima hormiga seleccione el camino. Por lo tanto, cuantas más hormigas viajan por un camino, éste se convierte en un camino más atractivo para hormigas posteriores.

En [Yu et al., 2009] se propone una optimización de colonias de hormigas (IACO), que posee una estrategia para actualizar el aumento de feromonas, llamada estrategia de peso de hormiga, y una operación de mutación. En la implementación una hormiga individual simula un vehículo y su ruta se construye seleccionando clientes de forma incremental hasta que todos los clientes han sido visitados. Los clientes que ya fueron visitados por una hormiga o violaron sus restricciones de capacidad, se almacenan en la lista de clientes inviables (tabú). La toma de decisiones sobre la combinación de clientes se basa en una regla probabilística que tiene en cuenta tanto la visibilidad como la información de las feromonas. De éste modo se generan las soluciones iniciales del problema y posteriormente sobre éstas mismas se aplica una mutación.

4.4.3.6 Enjambre de Luciérnagas

En [Asma et al., 2021] se propone una variante mejorada de CVRP-FA, un algoritmo de luciérnaga híbrido cooperativo (CVRP-CHFA) con múltiples poblaciones de algoritmos de luciérnaga (FA). Cada FA está hibridado con dos tipos de búsqueda local (es decir, 2-opt mejorado como búsqueda local y 2-h-opt como operador de mutación) y operadores genéticos. Los algoritmos propuestos se comunican de vez en cuando para intercambiar algunas soluciones (luciérnagas). La idea clave de este cambio es que la mutación se aplica ocasionalmente según una probabilidad predefinida (tasa de mutación). Al contrario de CVRP-FA, el algoritmo propuesto acepta todas las soluciones producidas por este cambio. Si se mejora la solución, el algoritmo intensifica su búsqueda en las áreas prometedoras; de lo contrario, diversifica el espacio de búsqueda y sale de los mínimos locales. En segundo lugar, los conceptos de modelo de isla cooperativa se utilizan para mantener más la diversidad, evitar quedar atrapado en mínimos locales y acelerar la convergencia.

La técnica de luciérnaga utilizada en [Yesodha y Amudha, 2019] intenta mejorar la calidad de una solución inicial mediante el uso de un algoritmo de ahorro de Clarke y Wright y una técnica de búsqueda local, y su rendimiento es comparado con el algoritmo de luciérnaga estándar y luciérnaga mejorada.

La característica principal de las luciérnagas es su luz intermitente y tiene dos funciones básicas, es decir, fascinar a los compañeros de apareamiento y también advertir a los depredadores potenciales. El movimiento de una luciérnaga *i* es atraído hacia una luciérnaga más brillante *j*. La agrupación de los clientes al depósito se realiza en la fase de asignación. En la fase de asignación, primero se calcula la distancia entre a) el depósito a los clientes y b) de clientes a clientes utilizando las coordenadas *x* e *y*. De sus coordenadas se determinan la intensidad de la luz y el coeficiente de absorción.

A continuación, en función de la mayor intensidad de luz, los clientes se clasifican en orden descendente y se asignan a los depósitos. Si no se encuentran otros más brillantes, los clientes se clasificarán en orden aleatorio y se asignarán en sus respectivos depósitos. La fase de enrutamiento se calcula utilizando el algoritmo de [Clarke, y Wright, 1964], mejorado aún más por la técnica intra e inter ruta.

4.4.3.7 Enjambre de abejas

En [Szeto et al., 2011] se propone la implementación de un algoritmo de colonia de abejas artificiales y también se propone una versión mejorada de ésta heurística. El algoritmo pertenece a una clase de algoritmos evolutivos que se inspiran en el comportamiento inteligente de las abejas melíferas para encontrar fuentes de néctar alrededor de sus colmenas. En el algoritmo propuesto, las abejas se dividen en tres tipos: abejas obreras, espectadoras y exploradoras. Las abejas obreras son responsables para aprovechar las fuentes de alimentos disponibles y recopilar la información necesaria. También comparten la información con los espectadores y los espectadores seleccionan las fuentes de alimentos existentes para seguir explorando. Cuando la abeja obrera abandona la fuente de alimento, la abeja obrera se convierte en un explorador y comienza a buscar un nuevo alimento fuente en las proximidades de la colmena. El abandono ocurre cuando la calidad de la fuente de alimento no mejora después de realizar un número máximo permitido de iteraciones.

El algoritmo es iterativo y comienza generando soluciones aleatorias como fuentes de alimentos y asignando a cada abeja obrera como fuente de alimento. Luego, durante cada iteración, cada obrera encuentra una nueva fuente de alimento cerca de su asignado originalmente (o antigua) fuente de alimento (usando un operador de vecindario). Luego se evalúa el néctar (aptitud) de la nueva fuente de alimento. Si la nueva fuente de alimento tiene más néctar que la anterior, entonces la vieja fuente es reemplazada por la nueva. Después de que todas las abejas obreras hayan terminado con el proceso de explotación anterior, comparten el néctar (información de las fuentes de alimentos) con los espectadores.

4.4.3.8 Hibridaciones

En [Shih et al., 2009] se aplica un algoritmo híbrido de Recocido Simulado y Búsqueda Tabú combinado con la búsqueda local, el algoritmo comienza con 8 parámetros definidos, los parámetros que deben definirse para Recocido Simulado y 2 más que son la permanencia mínima y máxima del movimiento tabú, mediante éstos mecanismos combinados el cliente será visitado de la siguiente manera. El primer vehículo sale antes del depósito y regresa al depósito después de visitar a los

clientes en esa ruta. Otros vehículos salen del depósito en orden y la disposición de la ruta de cada vehículo se lleva a cabo en cada proceso de iteración, hasta que cada cliente tenga un vehículo visitado. Al finalizar la implementación de éste mecanismo se agrega una búsqueda local para mejorar la solución.

[Berger y Barkaoui, 2003] propone un nuevo algoritmo genético híbrido para abordar el CVRP. El esquema básico consiste en la evolución simultánea de dos poblaciones de soluciones para minimizar la distancia total recorrida utilizando operadores genéticos que combinan variaciones de conceptos clave inspirados en técnicas de enrutamiento y estrategias de búsqueda utilizadas para una variante temporal del problema para proporcionar una guía de búsqueda adicional mientras se equilibra la intensificación y la diversificación.

Éste enfoque consiste en la evolución simultánea de dos poblaciones de soluciones (Pop1, Pop2), mientras se intercambia un cierto número de individuos (migración) al final de una nueva generación. El proceso evolutivo se repite hasta que se cumple una condición de parada predefinida.

El algoritmo genético de estado estable propuesto recurre a poblaciones superpuestas para garantizar el reemplazo de la población para Pop1 y Pop2. El procedimiento de selección es estocástico y sesgado hacia las mejores soluciones utilizando un esquema simple de rueda de ruleta.

En [Altabeeb et al., 2019] se propone un algoritmo CVRP-FA, que es el resultado de una hibridación de un algoritmo de Luciérnaga con dos procedimientos de búsqueda local. La función objetivo es minimizar la distancia total de todas las rutas en la luciérnaga correspondiente.

Dentro del algoritmo se utilizaron dos tipos de técnicas de mutación para mantener la diversidad de las luciérnagas y evitar que las luciérnagas del próximo movimiento converjan en mínimos locales. Según una determinada probabilidad de mutación, cada luciérnaga de la población puede modificarse para producir una nueva luciérnaga a través de dos tipos de mutación o sólo uno de ellos. En la primera mutación, se seleccionan al azar dos clientes de la misma ruta y se intercambian. En la segunda mutación, se seleccionan dos clientes al azar de dos rutas aleatorias diferentes y se intercambian.

[Yurtkuran y Emel, 2010] Nos presenta un nuevo algoritmo similar al electromagnetismo, híbrido. El algoritmo similar al electromagnetismo (EMA) está basado en población, el algoritmo imita el mecanismo de atracción-repulsión entre partículas cargadas en un campo electromagnético. En la metodología EMA, cada partícula representa una solución y lleva una cierta cantidad de carga que es proporcional a la calidad de la solución. Las soluciones se definen a su vez por vectores de posición que dan las posiciones reales de las partículas en un espacio multidimensional. Además, los valores de la función objetivo de las partículas se calculan basándose en estos vectores de posición. Cada partícula ejerce fuerzas de repulsión o atracción sobre otra población de miembros y una fuerza resultante sobre una partícula se utiliza para actualizar su posición. La idea detrás de la metodología EMA es mover partículas hacia la solución óptima ejerciendo atracción o fuerzas de repulsión.

4.4.3.9 Otras metaheurísticas

En [Hosseinabadi et al., 2017] se nos muestra un algoritmo de búsqueda local de emulación gravitacional, éste se basa en la ley de la gravedad y un grupo interacciones. El algoritmo propuesto utiliza dos de los cuatro parámetros básicos de velocidad y fuerza gravitacional en física, basada en

los conceptos de búsqueda aleatoria y agentes de búsqueda, que son una colección de masas que interactúan entre sí según la gravedad newtoniana y las leyes del movimiento.

[Shin y Han, 2011] presentan un algoritmo heurístico basado en centroides, El algoritmo propuesto consta de tres fases: construcción de grupos, ajuste de grupos y establecimiento de rutas. En la fase de construcción del clúster (grupo), el nodo más lejano (cliente) se selecciona entre los nodos no agrupados como semilla para formar un grupo. La noción de centro geométrico de un grupo se introduce en este estudio para ser utilizada en la construcción de dicho grupo y en las fases de ajuste del clúster. En segundo lugar, los nodos se mueven al clúster más cercano utilizando la noción del centro de un grupo. Finalmente, las rutas reales se establecen aplicando un algoritmo del problema de viajante (TSP) en los grupos ó cluster's producidos en las fases uno y dos.

En [Alssager et al., 2020] se implementó un algoritmo de búsqueda de Cuco, la búsqueda Cuco (CS o Cuckoo Search) es una metaheurística popular basada en la estrategia reproductiva de la especie de aves: Cuco, en el estudio realizado en el artículo se implementó una CS híbrida con algoritmo de recocido simulado (SA). El parasitismo de la cría es una característica interesante del pájaro cuco, en el que una hembra de cuco pone sus huevos en otro nido del ave observada y deja que el ave huésped incubar y criar a los polluelos de cuco. El cuco adapta sus huevos como un ave huésped en cuanto a patrones, formas y colores para aumentar la probabilidad de que nazca un nuevo cuco y la reducción de la probabilidad de que el ave huésped abandone los huevos [Yang y Deb, 2009].

El CS fue desarrollado recientemente por Yang y Deb [Yang y Deb, 2009] y fue diseñado inicialmente para resolver funciones multimodales siguiendo las reglas de que cada cuco pone un huevo a la vez y selecciona un nido al azar, el mejor nido con huevo de mayor calidad puede pasar, las nuevas generaciones y el número de nidos de hospedadores es fijo y el huevo puesto por un cuco puede ser descubierto por el ave hospedante con una probabilidad En CS, los cucos se abstraen como entidades en las que cada uno tiene una solución asociada (es decir, un huevo) del problema de optimización que intentan colocar en un contenedor de solución (es decir, el nido del pájaro anfitrión).

Capítulo 5. Metodología de solución

5.1 Introducción (Esquema general)

En capítulos anteriores se habló sobre el algoritmo de Recocido Simulado, el problema que se aborda en esta tesis y algunos métodos utilizados anteriormente para su solución. Ahora continuaremos explicando la metodología utilizada para solucionar dicho problema. El siguiente esquema muestra cómo se abordó el problema en el algoritmo.

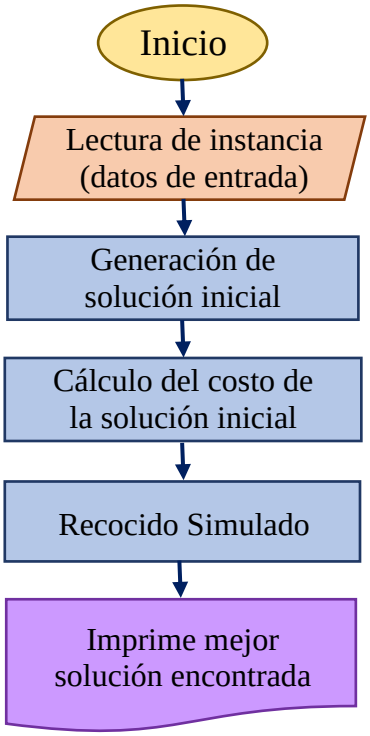


Figura 12. Esquema general de solución.

De manera general se implementó un algoritmo de Recocido Simulado para la solución del Problema de Enrutamiento de Vehículos con Capacidades Homogéneas, dicho algoritmo cuenta con un mecanismo para la generación de la solución inicial e implementa 3 vecindarios, que mediante distintas perturbaciones a la solución inicial generan nuevas soluciones, de este modo, también explicaremos el modo en que se calculó el costo de las soluciones.

5.2 Pseudocódigo del algoritmo propuesto

De acuerdo con el esquema de solución mostrado en el apartado 5.1, se presentará a continuación, el pseudocódigo del algoritmo propuesto, esto con el fin de ejemplificar más a fondo el modo en el que se implementó el algoritmo de recocido simulado sobre el problema.

Seleccionar un conjunto de rutas iniciales **Sactual** // **Dis(Sactual)** es la distancia total recorrida de las rutas generadas (función de costo)
Seleccionar un *criterio_de_parada* (**Temperatura_final**)
Seleccionar una temperatura inicial $T_0 > T_f > 0$
Seleccionar factor de decremento de la temperatura T_0
Seleccionar un número de iteraciones *nrep* // No. De ciclos de Metropolis
REPETIR
REPETIR
Generar mediante 3 vecindarios un nuevo conjunto de rutas *Svecina* ϵ N(*Sactual*)
Obtener la distancia: Dis(Svecina)
SI *Dis(Svecina) < Dis(Sactual)*
ENTONCES *S(actual)* *S(vecina)*
SINO Algoritmo de Metropolis
$\left(\exp - \frac{Dis(Svecina) - Dis(Sactual)}{T_0}\right) > random\ [0.1]$ Criterio de Boltzman
ENTONCES *Sactual* *Svecina*
FIN SINO
HASTA QUE cuenta_iteraciones=*nrep*

Como puede observarse en el pseudocódigo anterior, el algoritmo de Recocido Simulado implementado al problema de enrutamiento de vehículos, genera como soluciones actuales y vecinas: cadenas de números pertenecientes a las rutas, y (por medio de alteraciones a las rutas actuales) genera rutas vecinas, posterior a esto, obtiene la distancia total de cada solución y se estas se comparan, para así poder elegir el conjunto de rutas con menor distancia como la nueva solución actual.

A la solución vecina que resulta no tener una distancia menor que la solución actual se le asigna un porcentaje de aceptación de acuerdo al criterio de Boltzman, evitando así el estancamiento del algoritmo en óptimos locales, y permitiendo continuar generando soluciones vecinas y rutas distintas.

Es importante señalar que en el pseudocódigo mostrado anteriormente no se agregó la comparación de la capacidad de los vehículos, es decir, recordando el problema, todos los vehículos cuentan con una capacidad que, aunque es idéntica para todos, es necesario verificar al momento de la creación del cualquier conjunto de rutas, esto para cumplir con las restricciones del problema, el proceso de generación de rutas iniciales y vecinas se explicará con mayor detalle en apartados posteriores.

5.3 Formato de datos de entrada (Instancias)

Para comenzar la solución del problema es necesario inicialmente tener los datos de dicho problema, así, el primer paso fue realizar la lectura de las instancias de prueba, las instancias contienen los datos que el algoritmo utiliza para generar soluciones, para éste algoritmo utilizamos instancias del Benchmark de Augerat et al. De los grupos de instancias A y B, dichas instancias contienen, en un archivo .VRP, el formato siguiente:

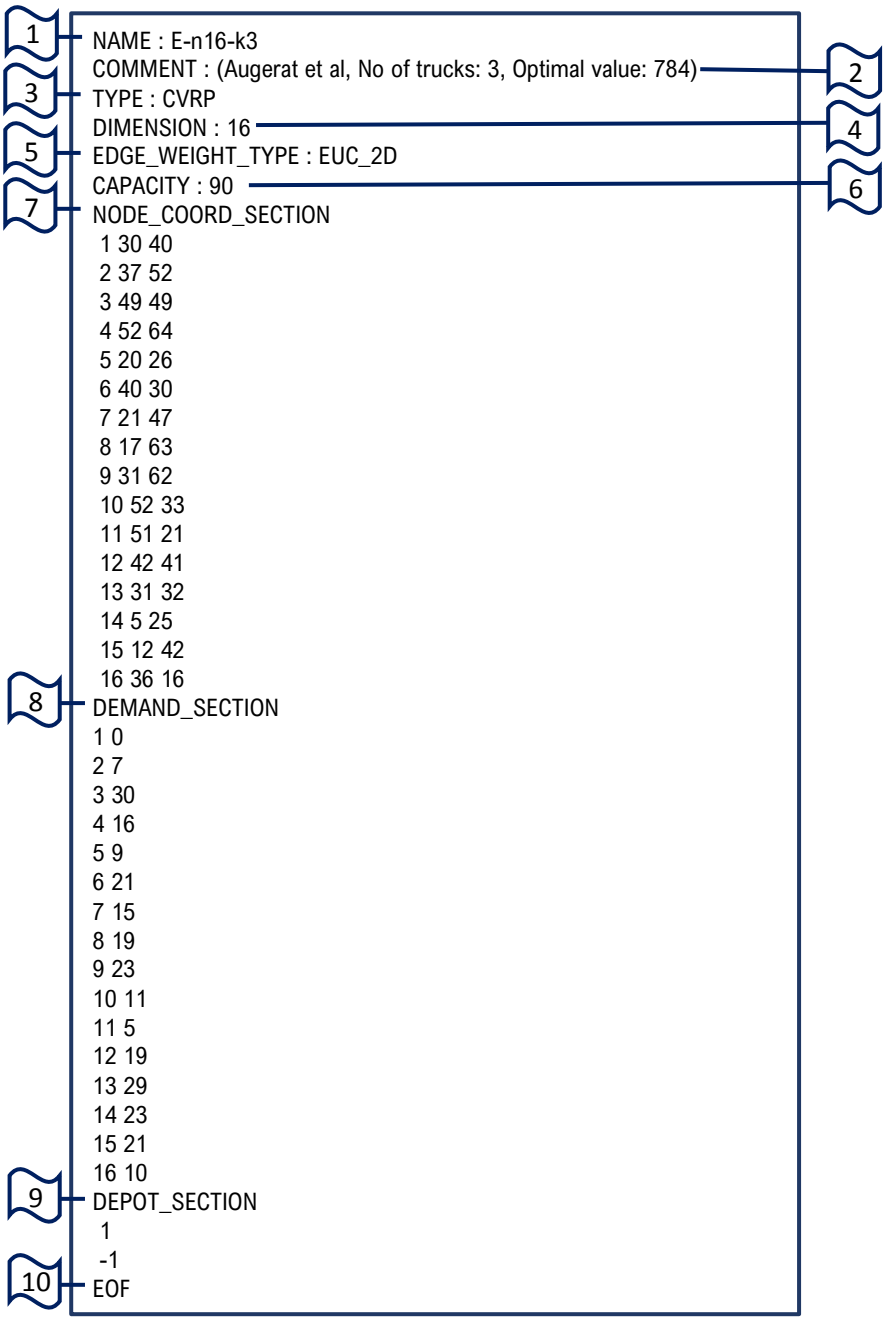


Figura 13. Formato de instancias

Donde:

1) Nombre de la instancia: Nos dice en orden: "Grupo_de_la_instancia -n número_de_clientes -k número_de_rutas_máximo_a_generar".
2) Comentario: contiene el nombre del Benchmark al que pertenece la instancia, número máximo de rutas a generar y el valor óptimo ó mejor costo de la instancia.
3) Tipo: Es la variante del problema para el cuál fue diseñado la instancia, en este caso CVRP.
4) Dimensión: Número de clientes de la instancia.
5) Nos indica el tipo de datos de la instancia, para este caso: Euclidiana de 2D, esto significa que la instancia tiene 2 coordenadas: "x" y "y", con éstas y la fórmula de la distancia euclidiana será posible calcular las distancias entre clientes y depósito y por consecuente el costo total de las soluciones.

6) Capacidad: El dato contiene la capacidad total de los vehículos, recordemos que para todos los vehículos la capacidad es la misma.
7) Sección de coordenadas del nodo: en esta parte encontramos ya la ubicación de los clientes dada en el siguiente orden: de izquierda a derecha encontramos el número de cliente o nodo, coordenada x y coordenada y, con éstos datos podemos saber la ubicación de cada cliente y del depósito.
8) Sección de demanda: En esta parte se nos indica la demanda, de izquierda a derecha se nos dice el número de cliente o nodo seguido de su demanda.
9) Sección del depósito: Nos dice qué número de nodo corresponde al depósito, en éste caso el depósito es el nodo número 1, también podemos identificar al depósito revisando la lista de demandas ya que es el único nodo que tiene una demanda de "0" debido a que no debe ser abastecido.
10) Indica el fin de la instancia.

5.4 Diseño de solución inicial

Para generar la solución inicial del algoritmo es necesario tener en cuenta que dicha solución debe cumplir con el criterio de no sobrepasar la capacidad del vehículo. Como ejemplo de la manera en que el algoritmo genera la solución inicial tomaremos la instancia "E-n16-k3" del Benchmark de Augerat et al. perteneciente al grupo E, misma que fue mostrada en el apartado 4.1. La función que genera la solución inicial realiza lo siguiente:

Paso 1: El primer paso (después de almacenar los datos de entrada tal cuál vienen en la instancia) es comenzar a generar rutas sin tomar en cuenta el costo, para la solución inicial los datos que se utilizan son: el número cliente, demanda de cada uno y la capacidad del vehículo, por ahora las coordenadas de los clientes no serán tomadas en cuenta.

Primero generamos una ruta vacía cuya capacidad usada es 0 y no contiene aún ningún cliente asignado para abastecer. Nota: El nodo 1 con demanda 0 corresponde al depósito por lo que no será tomado en cuenta para generar la solución inicial. De manera visual podríamos representar lo descrito anteriormente de la siguiente forma

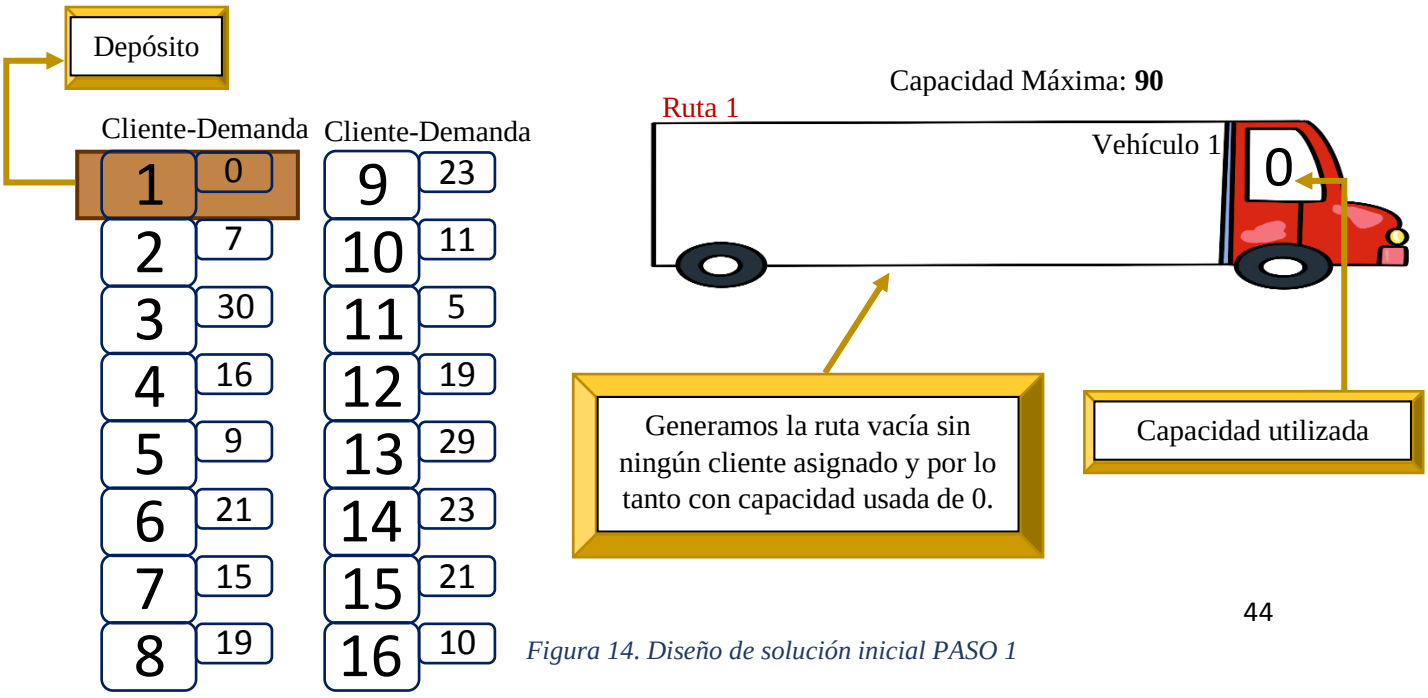


Figura 14. Diseño de solución inicial PASO 1

Paso 2: Elegimos un cliente al azar y lo asignamos a la ruta nueva, repetimos esto varias veces verificando la capacidad del vehículo hasta que ya no sea posible agregar a esta ruta el siguiente cliente elegido al azar.

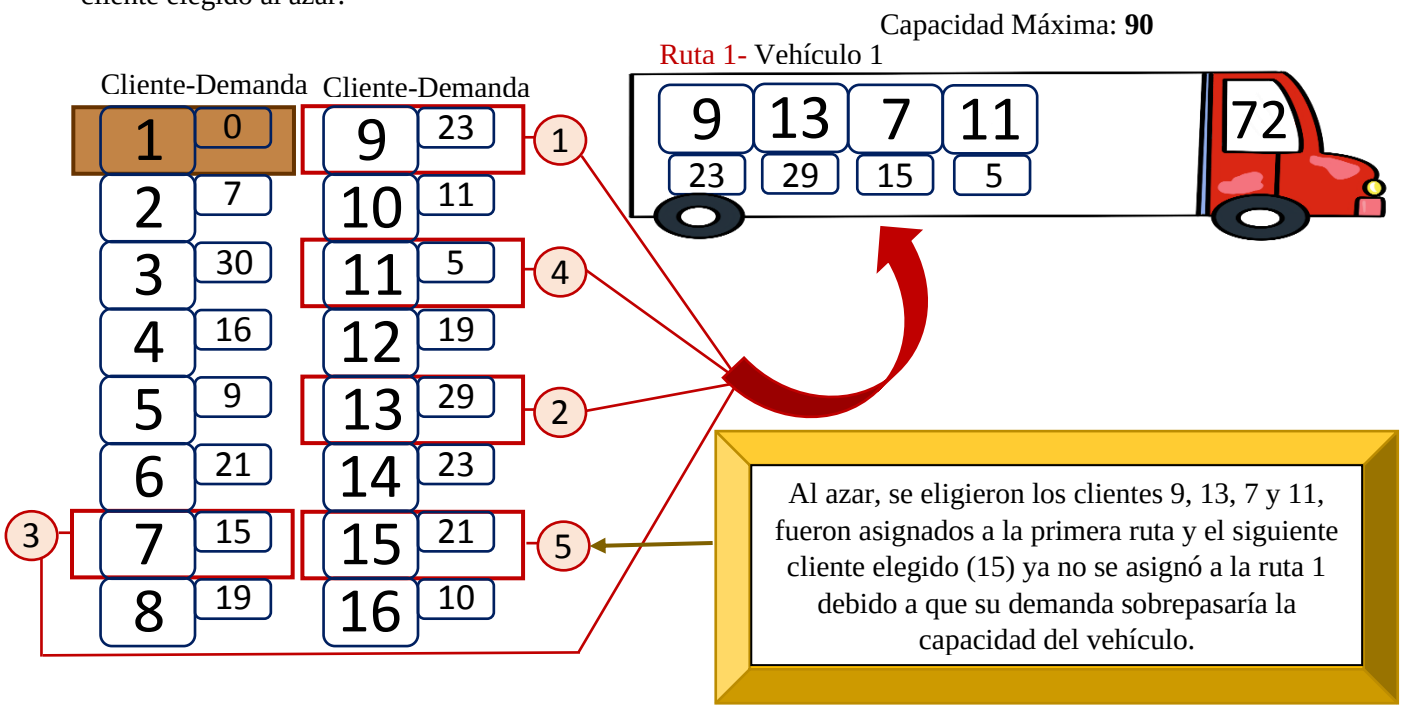


Figura 15. Diseño de solución inicial PASO 2

Paso 3: Una vez que la demanda del siguiente cliente elegido aleatoriamente ya no puede ser asignada a la ruta existente se genera una ruta vacía nueva, y se continúa el proceso de elegir clientes y asignarlos, es importante señalar que cada vez que se elija un cliente nuevo se verificará si puede ser asignado en cada una de las rutas existentes, es decir, al elegir un cliente antes de asignarlo a la ruta 4 debe revisarse si es posible asignarlo primero a la ruta 1, o a la 2, o a la 3 esto con el fin de aprovechar la capacidad de los vehículos y generar las menos rutas posibles.

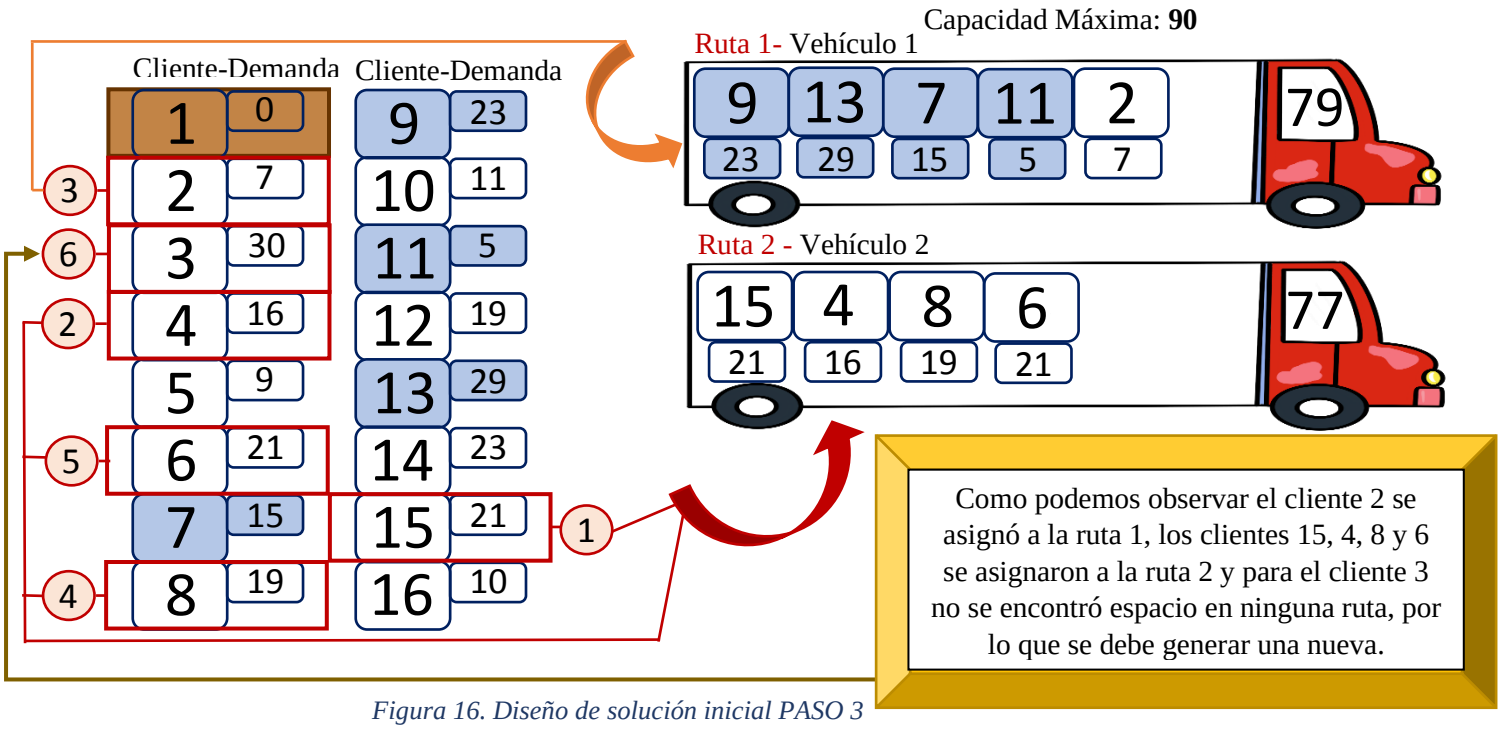


Figura 16. Diseño de solución inicial PASO 3

Se eligieron al azar los clientes 15, 4, 2, 8, 6 y 3, al elegir cada uno se verificó primero si podían agregarse a la ruta 1, de lo contrario se verificó si había espacio para asignarlos en la ruta dos, por esto el cliente 2 se asignó a la ruta 1, los clientes 15, 4, 8 y 6 se asignaron a la ruta 2 y para el cliente 3 (que fue el siguiente seleccionado) no se encontró espacio en ninguna ruta, por esto, se vuelve a repetir la creación de una ruta nueva y así se continúa repitiendo el proceso de verificación de capacidad, asignación de clientes a vehículos y generación de rutas nuevas vacías hasta que se hayan asignado todos los clientes a un vehículo.

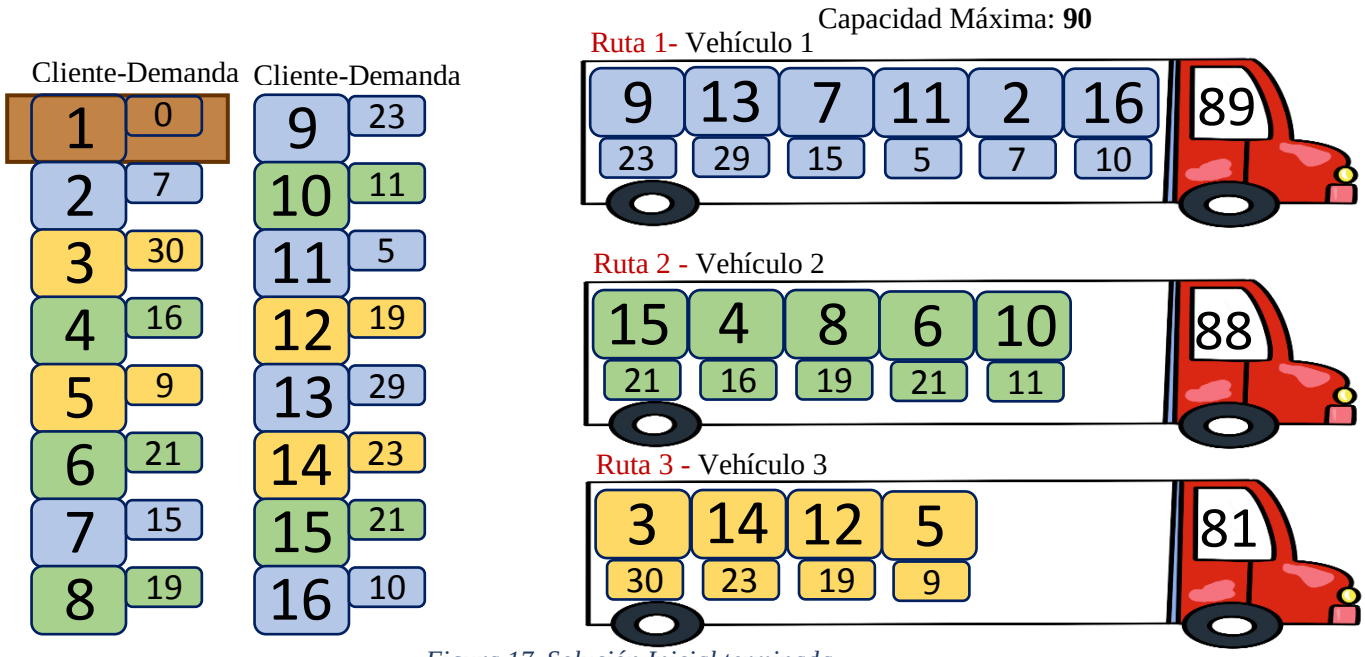


Figura 17. Solución Inicial terminada.

Con esto ya tendríamos generada la solución inicial, que en el caso del ejemplo consta de 3 rutas con un total de capacidad utilizada de 89, 87 y 81x respectivamente, hay que señalar que una de las restricciones del problema menciona que cada ruta debe iniciar y terminar en el depósito, por lo que a las rutas ejemplificadas anteriormente debe agregarse posteriormente al principio y al final el nodo número 1 perteneciente al depósito, dicha restricción no fue tomada en cuenta para generar la solución inicial debido a que el depósito no cuenta con una demanda, sin embargo es importante si tomar en cuenta la restricción al momento de calcular el costo de las soluciones, ya que, éste dato (el depósito) si suma distancia y por lo tanto influye en el aumento del costo.

5.5 Costo de las soluciones

Como se mencionó anteriormente el costo de la solución para el CVRP consiste en sumar las distancias existentes entre los nodos de todas las rutas generadas, para obtener esto se utiliza la fórmula de la distancia euclidiana, dicha fórmula nos permite obtener la distancia entre dos puntos, por lo que para obtener el costo de la ruta completa debe aplicarse dicha fórmula a cada par de puntos.

Para ejemplificar el modo en que se calcula la distancia entre dos puntos vamos a utilizar la solución inicial generada en el apartado anterior, tomaremos los dos primeros clientes asignados a la ruta 1 y por medio de la fórmula calcularemos la distancia euclidiana entre ellos.

Paso 1: Para calcular la distancia entre dos puntos lo primero es identificar los clientes (los dos puntos) de los cuáles queremos saber su distancia y conocer la ubicación de éstos, por lo que para

este paso deberemos tomar las coordenadas de los clientes, que como recordaremos, nos fueron dadas en la instancia como dato de entrada.

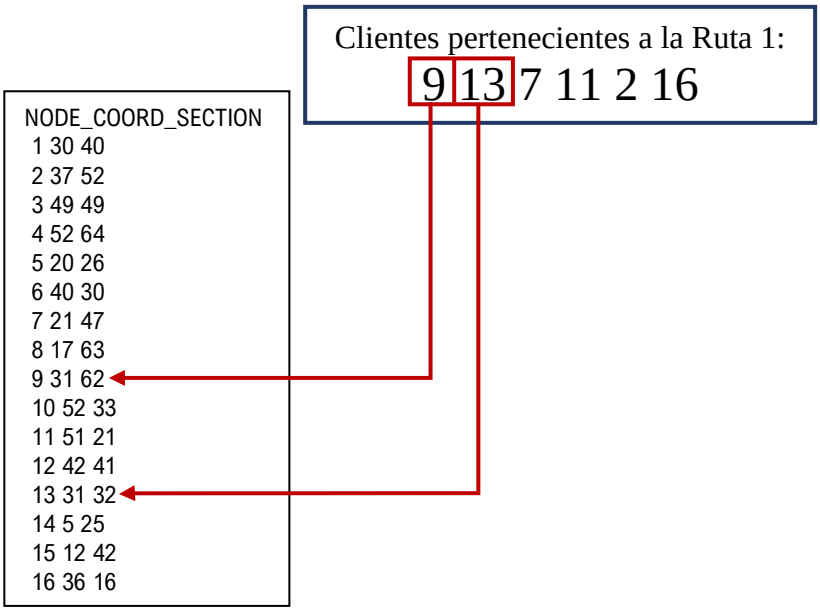


Figura 18. Coordenadas par de puntos

Paso 2: Ya teniendo las coordenadas de ambos puntos debemos identificar (de acuerdo a la fórmula) nuestras variables que identifican cada coordenada que en este caso las variables son: x1, x2, y1 y y2 correspondientes a las coordenadas de p1 (punto 1) y p2 (punto 2). Con esto podemos proceder a sustituir y desarrollar la fórmula que es la siguiente:

Punto 1	x_1	y_1
9	31	62
Punto 2	x_2	y_2
13	31	32

$$d_{(p_1,p_2)}=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

$$d_{(p_1,p_2)}=\sqrt{(31-31)^2+(32-62)^2}$$

$$d_{(p_1,p_2)}=\sqrt{0+(-30)^2}$$

$$d_{(p_1,p_2)}=\sqrt{900}$$

$$d_{(p_1,p_2)}=30$$

Figura 19.. Coordenadas de 2 clientes

De esta manera se obtiene la distancia entre cada par de puntos, este procedimiento debe repetirse hasta completar la suma de la distancia de toda la solución, es decir, de todas las rutas que la conforman, recordemos que cada ruta debe comenzar y terminar en el depósito así que en el algoritmo al calcular el costo de la solución también se agregó el cálculo de estas distancias, para así cumplir con la restricción del problema y obtener un correcto costo de las soluciones.

5.6 Vecindarios

Como se mencionó en el capítulo 2, conocemos como "vecindario" al conjunto de soluciones vecinas generadas realizando perturbaciones o modificaciones en cada iteración a partir de una solución inicial, dichos vecinos se forman realizando un intercambio entre los clientes, ya sean de rutas distintas o de la misma ruta. Una solución vecina es el resultado de realizar un intercambio de este tipo a la solución inicial, ya que con pequeñas modificaciones la solución y su costo pueden variar considerablemente, en cada iteración del algoritmo se realiza un intercambio de clientes y se calcula el costo de la nueva solución para que, por medio de la comparación de estos valores podamos evaluar si la modificación mejoró o empeoró la calidad

Los 3 Vecindarios (correspondientes a 3 tipos de movimientos distintos) aplicados en el algoritmo del trabajo presente son los siguientes:

5.6.1 Intercambio misma ruta

El primer mecanismo de perturbación empleado en el algoritmo es el intercambio de dos clientes de la misma ruta, para esto, se seleccionan dos clientes al azar (pertenecientes a la misma ruta) y se intercambian de puesto, tomando el primero el lugar del segundo y viceversa, con este cambio de orden, el costo de la solución también cambiará ya que se alteraría la distancia total de esa ruta y por consecuente el costo de la solución completa.

Este vecindario no altera el total de la capacidad de ruta ya que ambas demandas de los clientes estaban consideradas con anterioridad, por lo que solo requiere un cálculo nuevo del costo.

Para ejemplificar el vecindario tomaremos la ruta 1 generada en la solución inicial:

- Primero elegimos dos clientes al azar.
- Después realizamos el cambio de orden entre los dos, modificando así el total de la distancia de la ruta

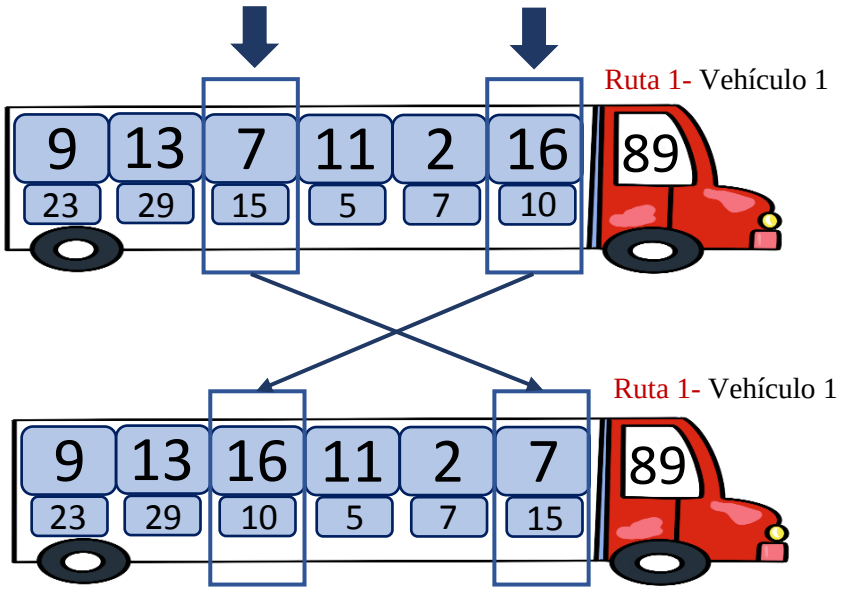


Figura 20. Intercambio misma ruta

5.6.2 Intercambio rutas distintas

La segunda vecindad empleada es similar al anterior con la variación de que las rutas son distintas, es decir, se realiza el intercambio de dos clientes o nodos, pero pertenecientes a rutas diferentes.

De igual manera se seleccionan dos clientes al azar (esta vez pertenecientes a rutas distintas) y antes de realizar el intercambio se verifica si este es factible, dicha factibilidad depende de que el cliente que se recibe no sobrepase la capacidad máxima del vehículo. Por lo que para éste cambio se requiere verificar las capacidades de ambos vehículos y recalcular el costo de la solución al finalizar el intercambio.

Para ejemplificar el vecindario tomaremos las rutas 1 y 2 generadas en la solución inicial:

- Primero elegimos dos clientes al azar.
- Verificamos que las capacidades de los vehículos no sean sobrepasadas, si esto ocurre se elige al azar otro par de clientes hasta que el cambio sea factible.
- Después realizamos el cambio de orden entre los dos clientes seleccionados, modificando así el total de la distancia de ambas rutas.

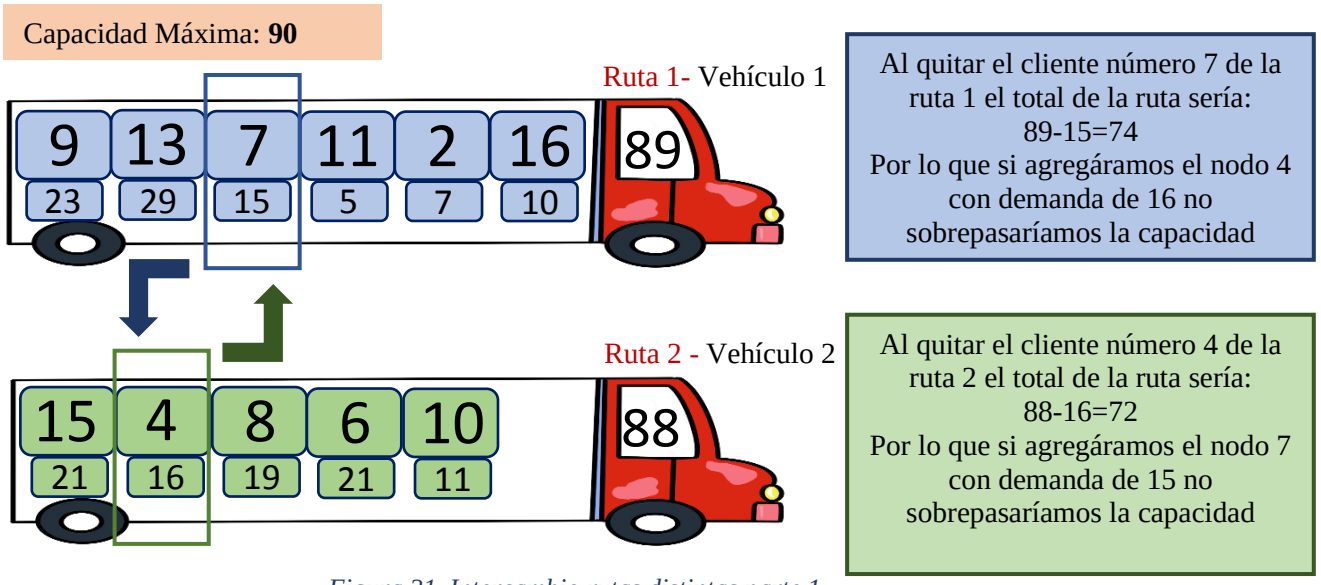


Figura 21. Intercambio rutas distintas parte 1

Al realizar el intercambio las rutas quedarían así:

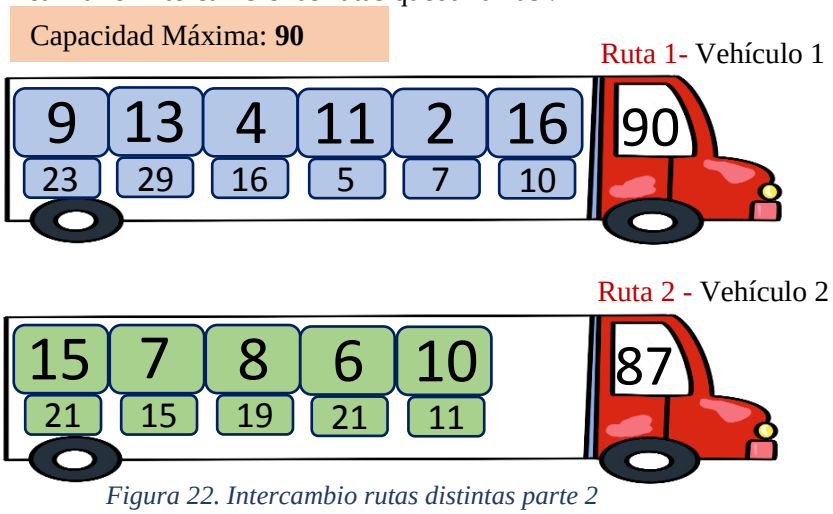


Figura 22. Intercambio rutas distintas parte 2

5.6.3 Reinserción

Para el tercer vecindario, se elige solamente 1 nodo al azar de cualquier ruta, y se verifica si este puede ser agregado en alguna otra (se recalcula la capacidad), no existe intercambio, solo se quita el nodo de la ruta a la que fue asignado anteriormente y se reinserta en otra, es importante resaltar que el cliente se puede insertar en cualquier orden de la ruta receptora: al inicio, en medio o al final, se inserta al azar en cualquier posición.

Para ejemplificar el vecindario tomaremos las rutas 1 y 3 generadas en la solución inicial:

- Primero se elige un cliente al azar, para el ejemplo se eligió el cliente 11 con capacidad de 5 asignado a la ruta 1.
- Ya elegido el cliente debe revisarse si existe espacio en alguna de las otras rutas, en la ruta 2 (de la solución inicial) con capacidad de 88 no es posible insertarlo, pero en la ruta 3 si, así que se realiza el cambio.
- La demanda del cliente se resta de la ruta donde estaba ubicado y se suma a la ruta que lo recibe modificando así los costos en ambas rutas.

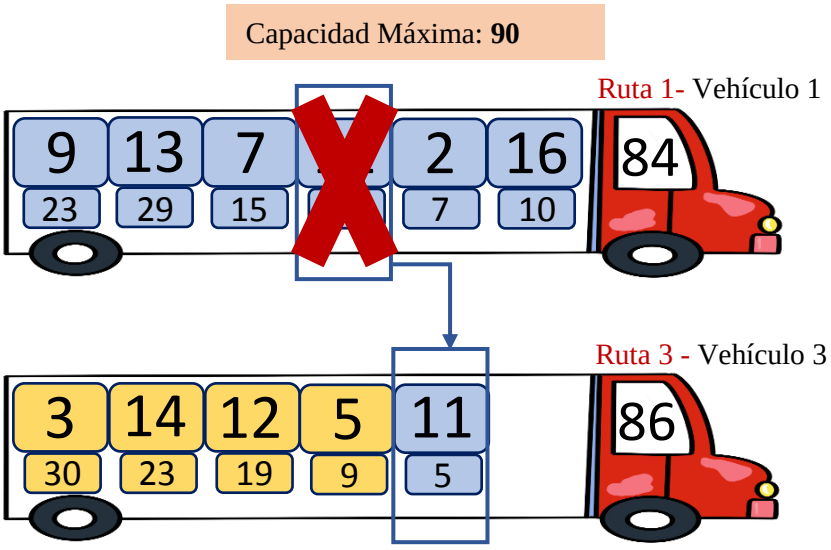


Figura 23. Reinserción parte 1

Al realizar la modificación las rutas quedarían así:

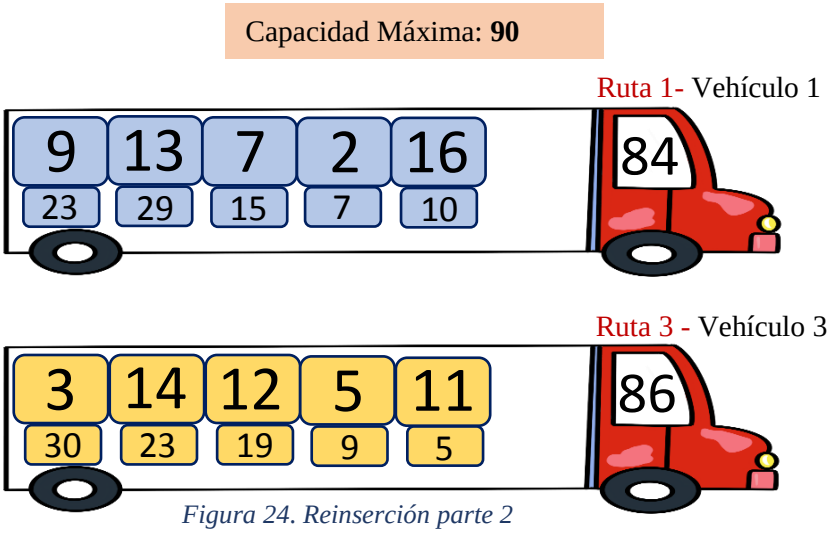


Figura 24. Reinserción parte 2

5.7 Representación de las soluciones

Una vez terminada la ejecución del recocido simulado, las soluciones obtenidas son representadas en un archivo de salida en formato .txt, (archivo que puede visualizarse a continuación), la siguiente representación es una solución obtenida de la instancia E-n16-k3 del Benchmark de Augerat et al., misma que fue explicada al inicio del capítulo y utilizada para los ejemplos de solución inicial y generación de vecindarios. Dicha solución es la siguiente:

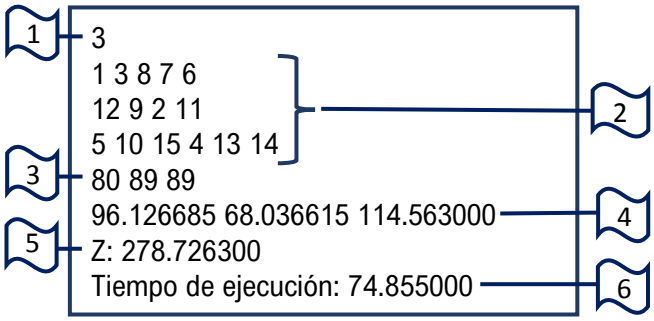


Figura 25. Representación de las soluciones

Dichos apartados representan:

1) Es el número total de rutas generado en la mejor solución encontrada.
2) Son los clientes asignados a cada ruta, representando una ruta por renglón.
3) Capacidad usada de los vehículos, correspondientes a cada una de las rutas.
4) Costo de cada ruta, es decir, distancia total que recorre cada ruta.
5) Costo total de la mejor solución encontrada (suma de los costos de las rutas).
6) Tiempo que tardó el algoritmo en encontrar esa solución.

5.8 Implementación del Recocido simulado

Como se observa en el esquema general de solución, el algoritmo de recocido simulado es una parte primordial para el programa, esto debido a que es donde la solución será mejorada.

La metaheurística de RS fue descrita anteriormente sin embargo puede ser implementada de distintas maneras dependiendo del problema a resolver, del tamaño de la instancia y de la persona que la implementa, por lo que a continuación se presenta un esquema referente a la aplicación del algoritmo en esta tesis.

Cabe señalar que una de las características que hace diferente la implementación del RS para este problema es que los vecindarios no se implementaron en orden comenzando siempre por la misma vecindad, sino que primero fue calculado el número de vecinos de un cliente elegido al azar, y posteriormente se eligió un vecino al azar, esto con el fin de agregar aleatoriedad al programa y no siempre comenzar con la aplicación del mismo vecindario.

A continuación, se presenta el esquema de la implementación del recocido simulado aplicado al presente problema.

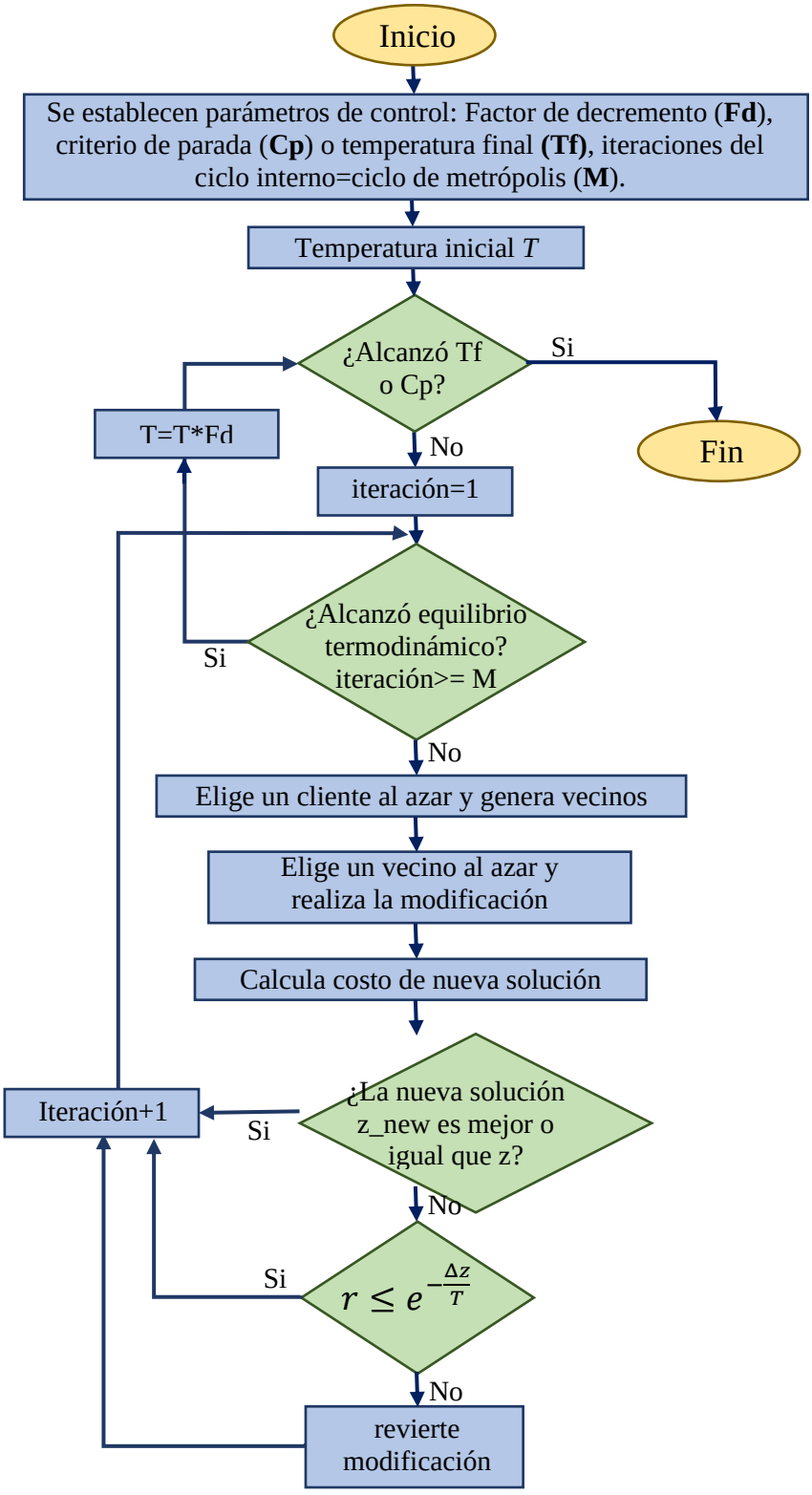


Figura 26. Implementación del Recocido Simulado

5.9 Sintonización

El proceso de sintonización consiste en ajustar los parámetros de control que se usan en la ejecución del algoritmo, con el objetivo de identificar aquellos valores que benefician más al desempeño del algoritmo y por consecuente generan mejores resultados.

La sintonización permite conocer la sensibilidad del programa al incrementar o decrementar valores de sus parámetros de control y permite a través de pruebas elegir los valores que mejoran la eficiencia del algoritmo, así como también permite saber si el incremento del tiempo mejora o no el resultado, esto con el fin de obtener buenas o muy buenas soluciones, pero siempre en un tiempo razonable.

Como se entiende, la sintonización es una parte importante para la implementación del algoritmo y para realizarla los parámetros que deben tomarse en cuenta son: Temperatura inicial (T_i), temperatura final (T_f), factor de decremento o *alfa*(α) y el número de iteraciones del ciclo interno o longitud de la cadena de Markov (*LCM*). Para realizar la sintonización se eligieron 6 instancias pertenecientes al grupo A del benchmark de Augerat et. al. (las mismas instancias serán utilizadas para los 4 parámetros) y cabe señalar que se realizó un total de 30 corridas del programa por cada valor de sintonización, esto con el fin de tener un mejor registro del comportamiento del algoritmo.

5.9.1 Temperatura inicial y temperatura final

Para realizar la sintonización de los parámetros mencionados no existe alguna regla específica ni rango de valores predeterminado que se vaya a tomar como punto de partida, así que para realizar la sintonización de la temperatura inicial y la temperatura final se analizó el porcentaje de aceptación, durante la sintonización de la temperatura inicial, se tomó como punto de referencia el valor de costo de la solución inicial de cada instancia, lo que hace que el valor de este parámetro dependa de la instancia que se está resolviendo.
El valor propuesto para la temperatura inicial se fijó a que sea igual al valor de la solución inicial. Para empezar a sintonizar el algoritmo, se tomaron como punto de partida los siguientes valores: La temperatura final se probó con los valores de 0.1, 0.01, 0.001 y 0.0001 mientras que para la temperatura inicial se hicieron pruebas con la división del costo de la solución inicial, con el fin de analizar el porcentaje de aceptación.

Tabla 5. Sintonización temperatura inicial y final

Porcentaje de aceptación									
Instancia	Optimo	Temperatura inicial				Temperatura final			
		Costo inicial	Costo inicial/2	Costo inicial/3	Costo inicial/4	0.1	0.01	0.001	0.0001
A-n37-k5	669	98.73	97.5	97.6	95.2	0.1	0.6	0.2	0.5
A-n39-k6	831	98.78	98.1	97	95.6	4	3.7	3.4	2.6
A-n45-k6	944	99.37	98.5	97.7	96.6	0.1	0	0.6	0
A-n53-k7	1010	99.41	99.1	98.2	97.5	0.4	0.2	0.9	0
A-n62-k8	1288	99.22	98.1	97.1	94.8	0.3	0.2	0.1	0.2
A-n80-k10	1763	99.76	99.3	98.9	96.8	0.5	0.3	0.1	0.4

En nuestras experimentaciones pudimos comprobar que el porcentaje de aceptación con los valores del costo inicial el porcentaje de aceptación es superior al 98% en todas las instancias probadas, por lo que se eligió dicho valor y en cuanto a la temperatura final observamos que con el valor 0.01 se encuentra una

5.9.2 Factor de decremento (*alfa*)

Las pruebas de alfa se realizaron tomando los siguientes valores: Para la temperatura inicial el valor ya establecido anteriormente fue el costo de la solución inicial, para la temperatura final se utilizó el valor 0.01 y para las iteraciones del ciclo interno se estableció un valor arbitrario en 1000.

Tabla 6. Sintonización factor de decremento

Factor de decremento (Alfa)											
Instancia	Optimo	0.9	Seg	0.99	Seg.	0.999	Seg.	0.9999	Seg.	0.99999	Seg.
A-n37-k5	669	682	0.246	670	2.44	669	24.21	669	197.76	669	2325.369
A-n39-k6	831	835	0.258	833	2.652	833	25.564	831	279.67	831	2562.538
A-n45-k6	944	975	0.271	946	2.651	951	25.908	944	295.64	944	3479.876
A-n53-k7	1010	1055	0.345	1021	3.374	1017	32.44	1015	1361.559	1011	3611.49
A-n62-k8	1288	1321	0.425	1313	4.001	1308	38.581	1311	329.96	1294	4362.83
A-n80-k10	1763	1843	0.546	1807	5.000	1768	48.441	1770	416.07	1764	5500.20

En base a los resultados obtenidos en la tabla anterior, el valor que se eligió para el factor de decremento o *alfa* fue el .9999, esto debido a que durante las evaluaciones se observó un comportamiento esperado en la relación entre el factor de decremento y la calidad de las soluciones. Como ya se mencionó, el valor final se fijó en 0.9999 dado que la mejora que representa el valor de 0.99999 a las soluciones es de pocas unidades, mientras que el tiempo de ejecución sigue incrementando.

5.9.3 Ciclo interno o longitud de la cadena de Markov

Para la sintonización del ciclo interno las pruebas se realizaron tomando los siguientes valores: Para la temperatura inicial se tomó el costo de la solución inicial, el factor de decremento se estableció en 0.9999 y la temperatura final en 0.01. Los valores probados para este parámetro son: 500, 1000, 1500 y 2000 iteraciones.

Tabla 7. Sintonización longitud de la cadena de Markov

Longitud de la cadena de Markov (*LCM*)									
Instancia	Optimo	500	Seg	1000	Seg.	1500	Seg.	2000	Seg
A-n37-k5	669	670	152.18	669	199.13	671	462.87	669	531.24
A-n39-k6	831	831	144.04	831	279.67	832	440.62	831	568.73
A-n45-k6	944	944	589.85	944	295.64	944	591.6	944	586.65
A-n53-k7	1010	1012	726.56	1015	1361.559	1014	43.09	1018	733.06
A-n62-k8	1288	1303	197.236	1311	329.96	1297	585.387	1299	746.668
A-n80-k10	1763	1786	248.467	1770	416.07	1779	709.628	1773	964.127

En base a los resultados, el valor elegido para el ciclo interno fue de 1000 iteraciones, esto debido a que los resultados son de buena calidad sin requerir un tiempo exponencial para obtenerlos, como se observa en la tabla las 2000 iteraciones también obtuvieron buenos resultados, un poco mejor a las 1000 iteraciones, sin embargo el tiempo creció casi al doble, por lo que se decidió tomar el valor de 1000 debido al factor del tiempo y a que el impacto en los resultados no es tan grande como para alcanzar más óptimos en otras instancias.

Capítulo 6. EXPERIMENTACIÓN Y ANÁLISIS DE RESULTADOS

En el presente apartado realizaremos un análisis del desempeño del algoritmo descrito en capítulos anteriores, para esto revisaremos los resultados del mismo mediante tablas que nos mostrarán los resultados óptimos obtenidos en las instancias o en su defecto la desviación estándar respecto al óptimo.

6.1 Descripción de Instancias:

En total el algoritmo fue probado con 50 instancias pertenecientes al Benchmark de Augeralt et al., 27 pertenecientes al grupo A y 23 al grupo B, respectivamente. Se realizaron un total de 30 corridas por instancia, cabe señalar que las instancias están conformadas por clientes que van desde 31 a 79 en el grupo A y 30 a 77 en el grupo B y en ambos grupos de instancias la localización de los clientes y su distribución es aleatoria.

Tabla 8. Organización de instancias Grupo A

Instancias Grupo "A"				
Instancia	Número de clientes	Número de rutas	Capacidad de los vehículos	Mejor solución
A-n32-k5	31	5	100	784
A-n33-k5	32	5	100	661
A-n33-k6	32	6	100	742
A-n34-k5	33	5	100	778
A-n36-k5	35	5	100	799
A-n37-k5	36	5	100	669
A-n37-k6	36	6	100	949
A-n38-k5	37	5	100	730
A-n39-k5	38	5	100	822
A-n39-k6	38	6	100	831
A-n44-k6	43	6	100	937
A-n45-k6	44	6	100	948
A-n45-k7	44	7	100	1146
A-n46-k7	45	7	100	914
A-n48-k7	47	7	100	1073
A-n53-k7	52	7	100	1017
A-n54-k7	53	7	100	1169
A-n55-k9	54	9	100	1073
A-n60-k9	59	9	100	1354
A-n61-k9	60	9	100	1035
A-n62-k8	61	8	100	1311
A-n63-k9	62	9	100	1619
A-n63-k10	62	10	100	1315
A-n64-k9	63	9	100	1414
A-n65-k9	64	9	100	1179
A-n69-k9	68	9	100	1166
A-n80-k10	79	10	100	1770

Tabla 9. Organización de instancias Grupo B

Instancias Grupo "B"				
Instancia	Número de clientes	Número de rutas	Capacidad de los vehículos	Mejor solución
B-n31-k5	30	5	100	672
B-n34-k5	33	5	100	788
B-n35-k5	34	5	100	955
B-n38-k6	37	6	100	805
B-n39-k5	38	5	100	549
B-n41-k6	40	6	100	829
B-n43-k6	42	6	100	742
B-n44-k7	43	7	100	909
B-n45-k5	44	5	100	751
B-n45-k6	44	6	100	678
B-n50-k7	49	7	100	741
B-n50-k8	49	8	100	1312
B-n51-k7	50	7	100	1032
B-n52-k7	51	7	100	747
B-n56-k7	55	7	100	707
B-n57-k7	56	7	100	1166
B-n57-k9	56	9	100	1598
B-n63-k10	62	10	100	1502
B-n64-k9	63	9	100	861
B-n66-k9	65	9	100	1319
B-n67-k10	66	10	100	1033
B-n68-k9	67	9	100	1275
B-n78-k10	77	10	100	1221

En las tablas anteriores puede observarse visualmente el conjunto de instancias con el que se trabajó, observando que:

- Las capacidades de todos los vehículos son las mismas (son homogéneas).
- El número de clientes y número de rutas que debe ser generado está indicado en el nombre la cada instancia.
- El número de clientes corresponde al número de nodos que contiene la instancia -1, constituyendo éste nodo al depósito.

6.2 Resultados del grupo "A"

La eficiencia del algoritmo de recocido simulado es medido por la calidad de la solución que produce, por esto debemos analizar 2 valores importantes: la desviación estándar (dispersión) y la diferencia en porcentaje con el valor óptimo.

Para la diferencia con el óptimo (dada por la diferencia de la mejor solución encontrada y la solución óptima) se utilizó la siguiente fórmula:

$$\Delta\ \% = \frac{M_{sol}}{V_{opt}}$$

En la tabla siguiente se muestran los resultados obtenidos donde las soluciones resaltadas en color naranja muestran un óptimo alcanzado y el tiempo esta medido en segundos.

Tabla 10. Resultados grupo A

Resultados grupo "A"							
Instancia	Optimo	Recocido simulado			Desviación Std.	Dif. con el opt.	Tiempo promedio
		El mejor	El peor	Promedio	**σ**	**Δ %**	
A-n32-k5	784	784	785	784.03	0.1825742	0%	178.77233
A-n33-k5	661	661	661	661	0	0%	177.89233
A-n33-k6	742	742	742	742	0	0%	174.21533
A-n34-k5	778	778	786	779.86	2.5560386	0%	183.45933
A-n36-k5	799	799	805	799.63	1.5196037	0%	193.047
A-n37-k5	669	669	672	669.43	0.6789106	0%	200.61367
A-n37-k6	949	949	955	950.8	2.0068847	0%	195.476
A-n38-k5	730	730	734	730.63	1.0980652	0%	196.54
A-n39-k5	822	822	825	822.90	1.3222238	0%	204.42333
A-n39-k6	831	831	835	833.03	0.4901325	0%	218.32833
A-n44-k6	937	937	937	937	0	0%	227.87833
A-n45-k6	944	948	980	965.86	8.7246513	0.42%	228.38153
A-n45-k7	1146	1146	1152	1147.23	1.478194	0%	242.579
A-n46-k7	914	914	917	914.20	0.6102572	0%	247.40267
A-n48-k7	1073	1073	1094	1074.43	4.0401718	0%	256.02833
A-n53-k7	1010	1017	1025	1019.80	1.6060231	0.69%	278.13233
A-n54-k7	1167	1169	1180	1173.93	2.7156422	0.17%	341.5181
A-n55-k9	1073	1073	1084	1073.90	2.1229453	0%	291.258
A-n60-k9	1354	1354	1371	1360.36	2.7975769	0%	323.388
A-n61-k9	1034	1035	1041	1036.83	1.913353	0.09%	300.88533
A-n62-k8	1288	1311	1323	1316.13	3.0369374	1.78%	333.533
A-n63-k9	1616	1619	1635	1630.16	3.0522274	0.18%	326.75433
A-n63-k10	1314	1315	1326	1319.86	2.6747005	0.07%	335.86867
A-n64-k9	1401	1414	1445	1424.06	8.8197792	0,92%	335.05933
A-n65-k9	1174	1179	1190	1184.43	3.0021065	0.42%	393.9068
A-n69-k9	1159	1166	1192	1174.06	6.1919488	0.60%	363.007
A-n80-k10	1763	1770	1805	1789.93	7.5060129	0.39%	421.38833

En la tabla anterior 15 resultados lograron igualar el valor óptimo reportado y para el resto de las soluciones el porcentaje de diferencia con el óptimo está entre 0.07% a 1.78%.

La relación entre los resultados obtenidos y los valores óptimos de cada instancia se pueden apreciar en la siguiente gráfica, donde la línea naranja corresponde al resultado optimo, la línea verde a la mejor solución alcanzada por el presente algoritmo, la línea azul al promedio y la línea gris a la peor solución alcanzada para cada instancia.

6.3 Resultados del grupo "B"

En la tabla 12 se muestran los resultados del recocido simulado en las instancias del grupo. Se obtuvieron 18 resultados lograron igualar el valor óptimo reportado y para el resto de las soluciones el porcentaje de diferencia con el óptimo está entre 0.09% a 1.12%.

Tabla 12. Resultados grupo B

Resultados grupo "B"							
Instancia	Optimo	Recocido simulado			Desviación Std.	Dif. con el opt.	Tiempo promedio
		El mejor	El peor	Promedio	**σ**	**Δ %**	
B-n31-k5	672	672	675	672.30	0.7497126	0%	168.93767
B-n34-k5	788	788	788	788	0	0%	186.412
B-n35-k5	955	955	959	955.13	0.7302967	0%	196.76067
B-n38-k6	805	805	806	805.30	0.4660916	0%	211.02867
B-n39-k5	549	549	549	549	0	0%	210.56333
B-n41-k6	829	829	848	829.70	3.4755302	0%	216.77367
B-n43-k6	742	742	746	742.60	1.2205143	0%	228.30333
B-n44-k7	909	909	923	910.76	3.9972692	0%	238.137
B-n45-k5	751	751	758	751.30	1.2905492	0%	230.47567
B-n45-k6	678	678	687	682.93	3.139661	0%	240.6861
B-n50-k7	741	741	741	741	0	0%	272.79167
B-n50-k8	1312	1312	1319	1314.16	2.2296332	0%	295.10577
B-n51-k7	1032	1032	1040	1034.80	2.3103963	0%	274.10737
B-n52-k7	747	747	748	747.06	0.2537081	0%	281.167
B-n56-k7	707	707	714	712.26	1.4605935	0%	299.32
B-n57-k7	1153	1166	1274	1211.89	31.29964	1.12%	354.26059
B-n57-k9	1598	1598	1632	1604.4	10.529433	0%	310.81767
B-n63-k10	1496	1502	1547	1533.23	14.896617	0.40%	339.023
B-n64-k9	861	861	875	864.63	2.7603515	0%	344.1852
B-n66-k9	1316	1319	1337	1324.56	4.889703	0.22%	341.56033
B-n67-k10	1032	1033	1051	1039.33	3.906964	0.09%	355.57533
B-n68-k9	1272	1275	1295	1284.36	5.2816098	0.23%	360.81033
B-n78-k10	1221	1221	1270	1235.13	10.364839	0%	413.23767

Los resultados del algoritmo son buenos, proporcionando óptimos en la mayoría de las instancias probadas, y en las que no se logró obtener dichos valores los resultados están muy cerca, tanto que no sobrepasan un porcentaje del 1.12% de diferencia con el óptimo.

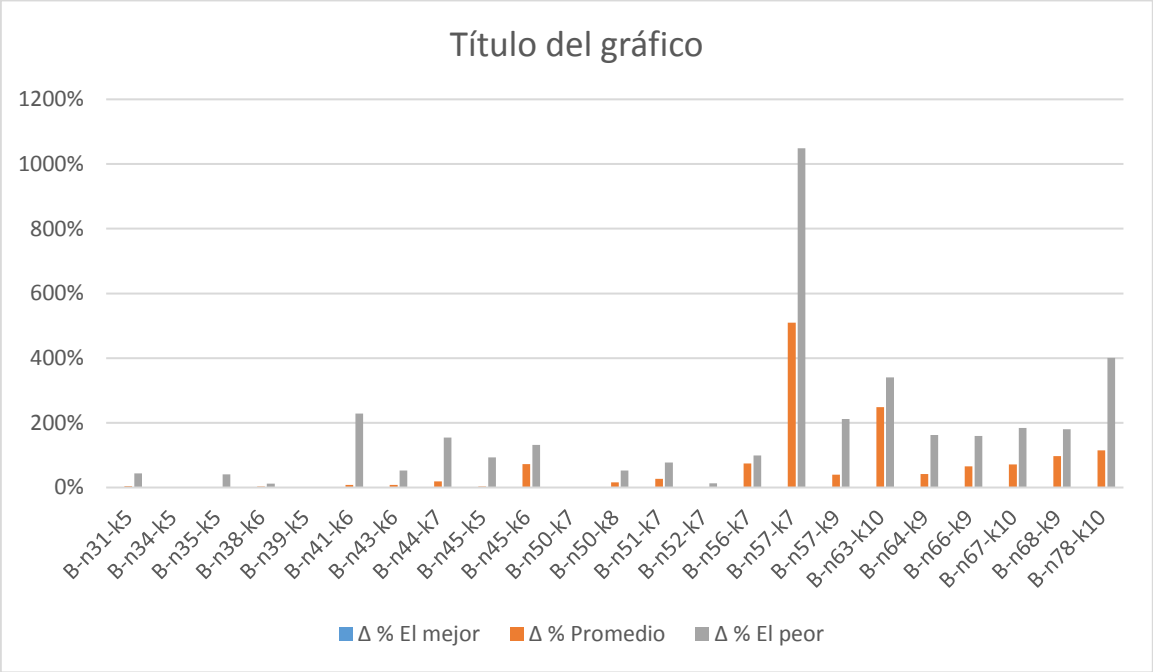


Figura 27. Gráfica de resultados del grupo B

De igual manera que con el grupo "A" los resultados del grupo "B" son buenos y en comparación con los óptimos, los resultados del algoritmo son bastante aproximados. El porcentaje de diferencia con el óptimo es menor al 1.12% lo que resulta ser una aproximación bastante buena.

De la gráfica se observa mejor (manera visual) que los resultados se encuentran muy próximos unos de otros en la mayoría de las instancias. Las líneas azul, verde y gris, correspondientes a los resultados del algoritmo, muestran una consistencia entre la mejor y peor solución para cada caso. Por su parte la línea de la mejor solución encontrada (verde) se encuentra en la mayoría de los resultados al tope de la línea que marca el resultado óptimo (azul) y en los resultados que no está al tope ésta se encuentra especialmente cerca de la línea azul, lo que significa una aproximación muy cercana al optimo.

6.4 Análisis Estadístico

Por ser una heurística la que se utilizó para la solución del problema, es necesario hacer un análisis estadístico, esto con el fin de analizar y comprender más a fondo la importancia de los resultados obtenidos, también, demostrar que el algoritmo implementado es distinto a otros algoritmos utilizados en la solución del mismo problema. Para este procedimiento llamaremos al algoritmo desarrollado en esta tesis: Recocido Simulado con Vecindad Variable por sus siglas: RSVV, con el fin de diferenciarlo de los demás algoritmos con los que se le va a comparar.

Para realizar un análisis estadístico es necesario primero, tener una hipótesis nula a refutar, esto con el fin de aclarar lo que queremos demostrar, en nuestro caso la hipótesis nula utilizada será la siguiente:

- **Hipótesis nula**: No existen diferencias en el comportamiento de RSVV vs las otras heurísticas

$$H_0: \overline{X_1} = \overline{X_2} = \cdots \overline{X_r}$$

- **Hipótesis alternativa:** Existen diferencias con el comportamiento de RSVV vs las otras heurísticas

$$H_1 = No\ todos\ son\ iguales$$

Una vez teniendo la hipótesis nula, realizamos una tabla comparativa, para éste análisis se incluyeron en la tabla 5 algoritmos distintos aplicados al mismo problema, dichos algoritmos son: SA, FFA, CENTROID, CHFA Y SWEEP. Como se observa en la tabla comparativa, para cada instancia fue añadido entre paréntesis un número indicando el Ranking por instancia, es decir, un número que, tomando en cuenta el problema de minimización, asigna un lugar del 1 al 5, indicando como el mejor resultado al que tiene el menor resultado obtenido para cada instancia.

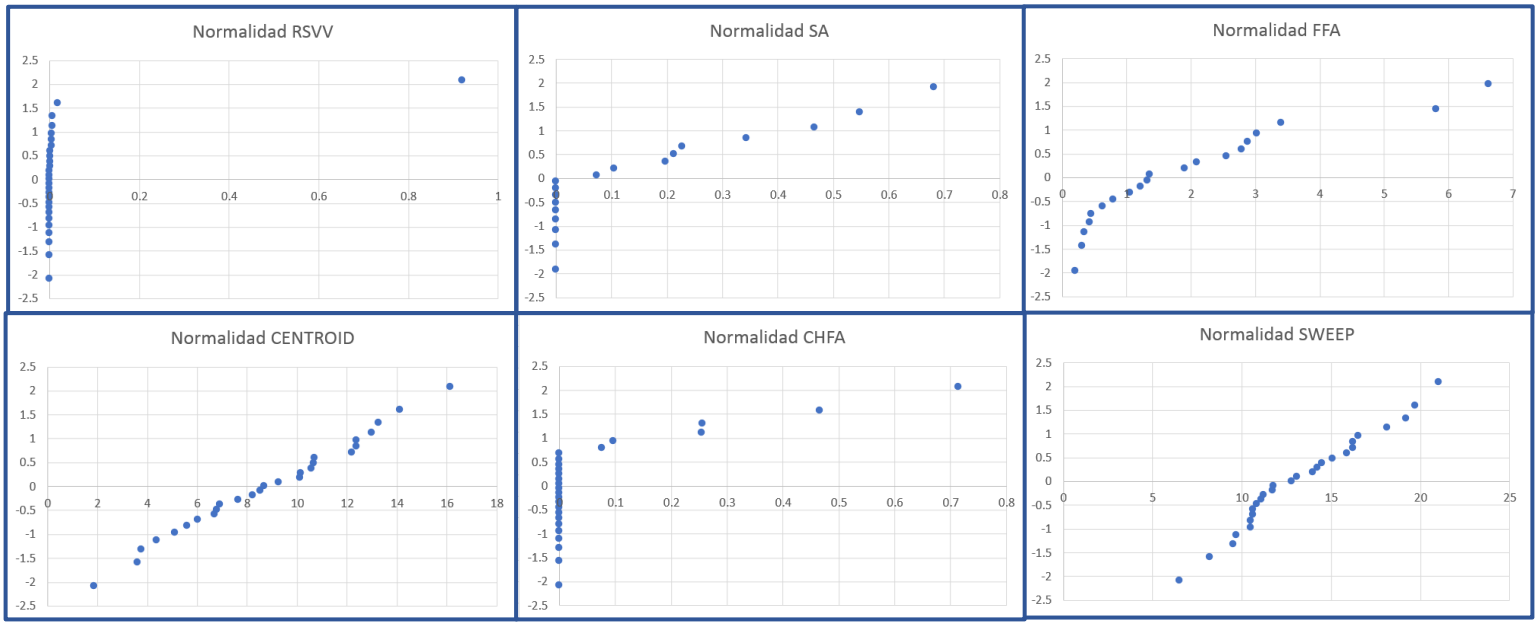
Al final de la tabla en la última fila se añadió también un Ranking general que nos muestra cuál algoritmo es el que obtiene mejores resultados, mostrando así que el propuesto en ésta tesis fue el mejor.

Instancia	OPTIMO	RSVV	SA	FFA	CENTROID	CHFA	SWEEP
A-n32-k5	784	0 (1)	0 (1)	- (6)	12.372 (5)	0.255 (3)	11.224 (4)
A-n33-k5	661	0 (1)	0 (1)	- (6)	10.136 (4)	0 (1)	19.213 (5)
A-n33-k6	742	0 (1)	0 (1)	- (6)	3.774 (4)	0 (1)	11.725 (5)
A-n34-k5	778	0 (1)	0 (1)	0.425 (4)	4.37 (5)	0 (1)	9.512 (6)
A-n36-k5	799	0 (1)	0 (1)	1.215 (4)	1.877 (5)	0 (1)	10.638 (6)
A-n37-k5	669	0 (1)	0 (1)	- (6)	13.004 (5)	0 (1)	9.716 (4)
A-n37-k6	949	0 (1)	0.105 (3)	- (6)	8.219 (4)	0 (1)	10.643 (5)
A-n38-k5	730	0 (1)	0.548 (3)	- (6)	12.192 (4)	0 (1)	19.726 (5)
A-n39-k5	822	0 (1)	0 (1)	2.088 (4)	5.109 (5)	0 (1)	18.127 (6)
A-n39-k6	831	0 (1)	0 (1)	0.795 (4)	6.017 (5)	0 (1)	16.245 (6)
A-n44-k6	937	0 (1)	0.213 (4)	0.208 (3)	10.672 (5)	0 (1)	16.542 (6)
A-n45-k6	944	0.004 (2)	- (6)	0.449 (3)	10.169 (4)	0 (1)	10.487 (5)
A-n45-k7	1146	0 (1)	0 (1)	0.314 (4)	12.391 (6)	0 (1)	11.78 (5)
A-n46-k7	914	0 (1)	- (6)	3.407 (3)	8.534 (4)	0 (1)	10.832 (5)
A-n48-k7	1073	0 (1)	- (6)	2.885 (3)	6.71 (5)	0 (1)	6.524 (4)
A-n53-k7	1010	0.007 (2)	0.198 (3)	1.049 (4)	10.594 (6)	0 (1)	10.495 (5)
A-n54-k7	1167	0.002 (2)	0.343 (3)	6.626 (5)	3.599 (4)	0 (1)	13.111 (6)
A-n55-k9	1073	0 (1)	- (6)	0.346 (3)	7.642 (4)	0 (1)	11.09 (5)
A-n60-k9	1354	0 (1)	0.074 (3)	1.905 (4)	5.613 (5)	0 (1)	16.248 (6)
A-n61-k9	1034	0.001 (1)	- (6)	2.552 (3)	16.151 (5)	0.097 (2)	14.507 (4)
A-n62-k8	1288	0.018 (1)	0.466 (2)	5.805 (4)	14.13 (5)	0.466 (2)	21.04 (6)
A-n63-k9	1616	0.002 (1)	- (5)	1.361 (2)	9.282 (3)	- (5)	12.809 (4)
A-n63-k10	1314	0.001 (1)	- (6)	0.623 (3)	6.925 (4)	0.076 (2)	15.906 (5)
A-n64-k9	1401	0.92 (2)	- (5)	- (5)	13.276 (3)	0.714 (1)	13.99 (4)
A-n65-k9	1174	0.004 (1)	0.681 (3)	3.021 (4)	8.688 (5)	0.256 (2)	15.077 (6)
A-n69-k9	1159	0.006 (2)	- (6)	2.793 (3)	10.699 (5)	0 (1)	8.197 (4)
A-n80-k10	1763	0.004 (2)	0.227 (3)	1.319 (4)	6.807 (5)	0 (1)	14.237 (6)

AVERAGE RANKING	1.222	3.259	4.148	4.593	1.37	5.111

Como se observa en la tabla, en la mayoría de las instancias el algoritmo propuesto tiene el mejor ranking, salvo en 6 instancias de 27.

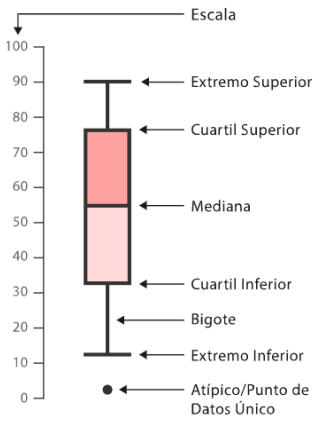
Una vez teniendo la tabla comparativa, se debe decidir si para realizar el análisis estadístico se utilizará un método "paramétrico" o "no paramétrico", para esto se deben analizar dos condiciones de aplicación importantes: la normalidad y la homocedasticidad. Primero se analizó la normalidad de los datos, para esto se generó una gráfica de normalidad para cada algoritmo mencionado en la tabla anterior, obteniendo los siguientes resultados:



Para que existe normalidad en los datos, al realizar las gráficas, en la distribución de los datos estos deben de ubicarse en la diagonal, después de observar las gráficas anteriores podemos decir que en los algoritmos anteriores NO EXISTE NORMALIDAD.

Teniendo la conclusión anterior, procedemos a analizar la homocedasticidad de los datos y para esto utilizamos gráficas de cajas y bigotes.

Las gráficas de cajas y bigotes son una manera muy clara de poder analizar de maera visual algunos grupos de datos, las cajas de las gráficas (que se observan a manera de cuadrados) se forman con los cuartiles y media de los datos analizados, las líneas que salen hacia arriba y debajo de dichas cajas son concidos como bigotes, éstos se utilizan para expresar variabilidad en los datos saliendo del valor de los cuartiles (fuera de los cuartiles) y en algunas ocaciones son colocados puntos fuera de las cajas, éstos representan valores atípicos.

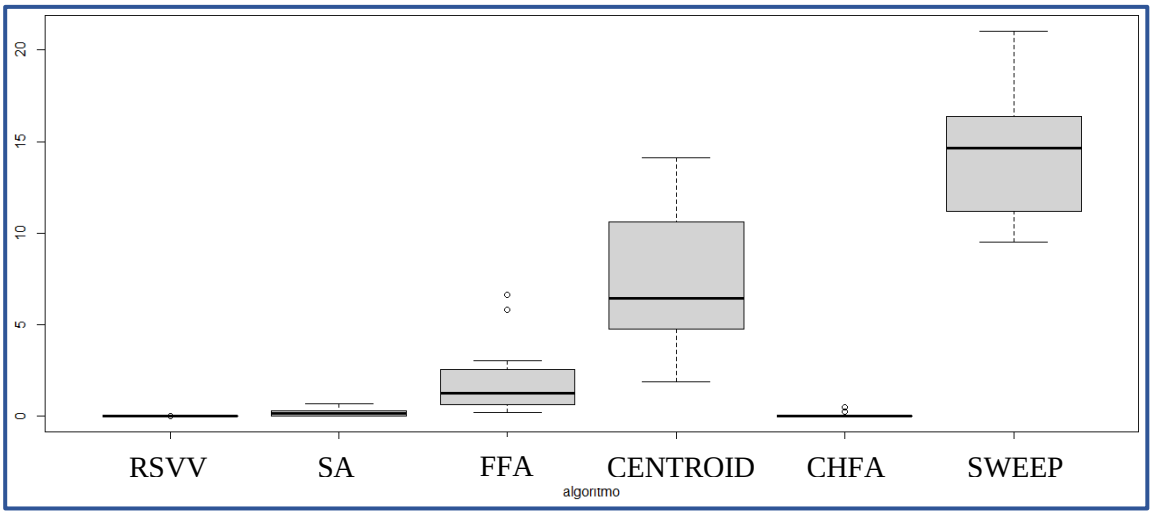


Las gráficas de cajas y bigotes nos ayudan a identificar de manera rápida y fácil:

- Si los datos son simétricos

- Si hay valores atípicos y cuáles son
- Cómo se agrupan los datos
- Y algunos valores clave como es: la mediana.

Teniendo en cuenta lo mencionado anteriormente podemos proceder a realizar la gráfica de cajas y bigotes para cada uno de los algoritmos con los que deseamos hacer el análisis estadístico, el resultado fue el siguiente:



Para que la homocedasticidad exista, las gráficas de cajas de cada algoritmo deben verse con la misma amplitud, es decir, deben observarse del mismo tamaño todas las cajas, analizando la tabla obtenida podemos notar fácilmente que las cajas no tienen ésta característica por lo que NO HAY HOMOCEDASTICIDAD. La homocedasticdad con respecto al algoritmo propuesto y el algoritmo CHFA se observa que es muy proxima, esto es debido a que los resultados obtenidos en ambos algoritmos son muy parecidos ya que la mayoria de dichos resultados coinciden con el óptimo conocido en la literatura.

Teniendo en cuenta los dos aspectos anteriores obtenemos las conclusiones: no existe normalidad en los datos, y no existe homocedasticidad, por lo que no puede aplicarse un método paramétrico para el análisis estadístico, así pues se eleigió el método no paramétrico de Kruskall-Wallis para realizar éste análisis estadístico.

Para realizar dicho procedimiento debemos realizar primero, basados en nuestra primera tabla comparativa, una tabla de rangos para poder aplicar posteriormente la fórmula del método, la tabla de rangos de nuestro análisis es la siguiente:

Instancia		SAVN	SA	FFA	CENTROID	CHFA	SWEEP
A-n32-k5		23	23	-	126	64	122
A-n33-k5		23	23	-	111	23	143
A-n33-k6		23	23	-	92	23	123
A-n34-k5		23	23	69	93	23	109
A-n36-k5		23	23	80	83	23	116
A-n37-k5		23	23	-	129	23	110
A-n37-k6		23	59	-	105	23	117

A-n38-k5		23	73	-	125	23	144
A-n39-k5		23	23	85	94	23	142
A-n39-k6		23	23	77	97	23	139
A-n44-k6		23	62	61	118	23	141
A-n45-k6		51	-	70	112	23	113
A-n45-k7		23	23	66	127	23	124
A-n46-k7		23	-	90	106	23	120
A-n48-k7		23	-	88	100	23	98
A-n53-k7		54	60	79	115	23	114
A-n54-k7		48.5	67	99	91	23	130
A-n55-k9		23	-	68	103	23	121
A-n60-k9		23	56	84	95	23	140
A-n61-k9		46.5	-	86	138	58	135
A-n62-k8		55	71.5	96	133	71.5	145
A-n63-k9		48.5	-	82	108	-	128
A-n63-k10		46.5	-	74	102	57	137
A-n64-k9		78	-	-	131	76	132
A-n65-k9		51	75	89	107	65	136
A-n69-k9		53	-	87	119	23	104
A-n80-k10		51	63	81	101	23	134
SUM RANK		951	793.5	1611	2961	851.5	3417

Una vez obtenida la tabla de rangos podemos proceder a aplicar la fórmula del método de Kruskall – Wallis, esto con el fin de validar el resultado y verificar si es capaz refutar la hipótesis nula o no.

Donde:
Z: es el estadístico de contraste
C: es el número de heurísticas a analizar
ni: número de observaciones en el ith del algoritmo
N: es el total de observaciones de todos los algoritmos
Ri: suma de los rangos en el ith del algoritmo, se ordenan los datos de menor a mayor y se re-enumeran de 1 a m sumándose estos valores por grupos de algoritmo
t: es el número de resultados repetidos
0.05 nivel de significancia. 5% de probabilidad de que los resultados se produzcan de forma aleatoria

Análisis estadístico Kruskal-Wallis solución

$$Z = \frac{\frac{12}{N(N+1)} \sum_{i=1}^{C} \frac{R_i^2}{n_i} - 3(N+1)}{1 - \frac{\sum(t^3 - t)}{N^3 - N}}$$

$$N = \sum n_i$$

$$Z = \frac{\frac{12}{145(145+1)} \left(\frac{951^2}{27} + \frac{793.5^2}{18} + \frac{1611^2}{20} + \frac{2961^2}{27} + \frac{851.5^2}{26} + \frac{3417^2}{27}\right) - 3(145+1)}{1 - \frac{(45^3 - 45) + (2^3 - 2) + (2^3 - 2) + (3^3 - 3) + (2^3 - 2)}{145^3 - 145}}$$

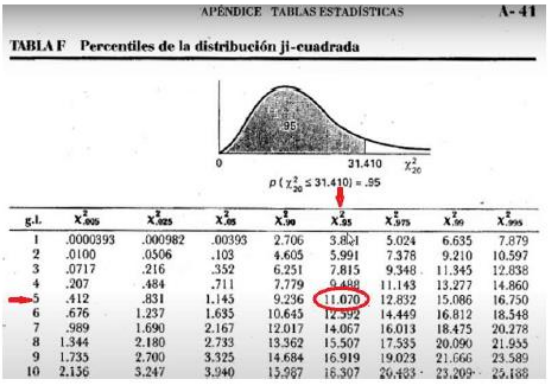
$$Z = \frac{\frac{12}{21170} \left(\frac{904401}{27} + \frac{629642.25}{18} + \frac{2595321}{20} + \frac{8767521}{27} + \frac{725052.25}{26} + \frac{11675889}{27}\right) - 3(145+1)}{1 - \frac{91122}{3048480}}$$

$$Z = \frac{\frac{12}{21170}\,(33496.3333 + 34980.125 + 129766.05 + 324723 + 27886.625 + 432440.333) - 3(145+1)}{1 - \frac{91122}{3048480}}$$

$$Z = \frac{557.3693 - 438}{1 - 0.02989096}$$

$$Z = \frac{119.3693}{0.97010904}$$

$$Z = 123.05$$

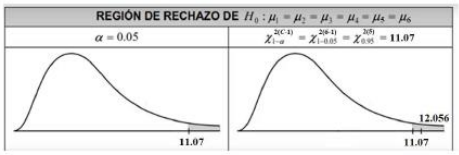


APÉNDICE TABLAS ESTADÍSTICAS A-41

TABLA F Percentiles de la distribución ji-cuadrada

g.l.	$\chi^2_{.005}$	$\chi^2_{.025}$	$\chi^2_{.05}$	$\chi^2_{.90}$	$\chi^2_{.95}$	$\chi^2_{.975}$	$\chi^2_{.99}$	$\chi^2_{.995}$
1	.0000393	.000982	.00393	2.706	3.841	5.024	6.635	7.879
2	.0100	.0506	.103	4.605	5.991	7.378	9.210	10.597
3	.0717	.216	.352	6.251	7.815	9.348	11.345	12.838
4	.207	.484	.711	7.779	9.488	11.143	13.277	14.860
5	.412	.831	1.145	9.236	11.070	12.832	15.086	16.750
6	.676	1.237	1.635	10.645	12.592	14.449	16.812	18.548
7	.989	1.690	2.167	12.017	14.067	16.013	18.475	20.278
8	1.344	2.180	2.733	13.362	15.507	17.535	20.090	21.955
9	1.735	2.700	3.325	14.684	16.919	19.023	21.666	23.589
10	2.156	3.247	3.940	15.987	18.307	20.483	23.209	25.188

Al no tener normalidad en los datos, el comportamiento de estos es el de una distribución ji-cuadrada.



El estadístico Z=123.05 está dentro de la región de rechazo que inicia en

$$x^{2(5)}_{0.95} = 11.07$$

Si Z >$x^{2(5)}_{0.95}$, H_0 se rechaza por existir diferencias en al menos un par de heurísticas, debido a que el estadístico de contraste obtenido que es 123.05 es mayor que 11.07.

Con los procedimientos anteriores comprobamos de manera general que la heurística desarrollada es distinta a las otras heurísticas analizadas, se comprobó que en al menos en 2 algoritmos existen diferencias, sin embargo, es necesario también demostrar que existe diferencia de nuestro algoritmo con cada uno de ellos, para esto debe hacerse una comparación individual para probar que la heurística es distinta a todas y cada una de ellas por separado.

Para realizar dicha comparación se realizó un análisis Post-Hoc Wilcoxon. Para dicho análisis debemos tener en cuenta que:

$W = min\,(W^+, W^-)$
• W+: suma de rangos con signo +
• W-: suma de rangos con signo -

Dichos datos serán utilizados en el desarrollo de la siguiente fórmula, en dicha fórmula está centrado el análisis Post- Hoc, con n <25 se comparará W con los valores de la tabla de Wilcoxon. Si W < Wc de tablas para esa n, se tienen diferencias significativas y H_0 se rechaza por lo que se acepta como verdadera la hipótesis alternativa.

- Con n<25
- $W^+ = \sum_{i-1}^{n} Z_i R_i(|D_i|)$
- $D_i = x_i - y_i$
- $z_i = \begin{cases} \mathbf{1}\ si\ D_i > 0 \\ 0\ si\ D_i < 0 \end{cases}$

$$Z=\frac{W(\frac{n(n-1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{12}}} \approx N(0,1)$$

De cada tabla comparativa se obtendrá un resultado que debe ser comparado con la tabla de valores críticos para la prueba de rango con signo, esto con el fin de saber si la hipótesis nula se deshecha o no. A continuación se presenta dicha tabla, en ella se señalan los primeros valores comparados con el resultado de la comparación de los algoritmos RSVV Y SA.

n	Unilateral α = 0.01 Bilateral α = 0.02	Unilateral α = 0.025 Bilateral α = 0.05	Unilateral α = 0.05 Bilateral α = 0.10
5			1
6		1	2
7	0	2	4
8	2	4	6
9	3	6	8
10	5	8	11
11	7	11	14
12	10	14	17
13	13	17	21
14	16	21	26
15	20	25	30
16	24	30	36
17	28	35	41
18	33	40	47
19	38	46	54
20	43	52	60
21	49	59	68
22	56	66	75
23	62	73	83
24	69	81	92
25	77	90	101
26	85	98	110
27	93	107	120
28	102	117	130
29	111	127	141
30	120	137	152

*Reproducida de F. Wilcoxon y R. A. Wilcox, *Some Rapid Approximate Statistical Procedures*, American Cyanamid Company, Pearl River, N. Y., 1964, con el permiso de la American Cyanamid Company.

W+ =	36
W- =	30
W=min(W+,W-)	30
W =	30

Instancia	RSVV				
A-n32-k5	0				
A-n33-k5	0				
A-n33-k6	0				
A-n34-k5	0				
A-n36-k5	0				
A-n37-k5	0	0	0	0	4.5
A-n37-k6	0	0.105	-0.105	0.105	9
A-n38-k5	0	0.548	-0.548	0.548	11
A-n39-k5	0	0	0	0	4.5

Así pues se procede a mostrar los resultados obtenidos del análisis Post – Hoc.

Para éste primer caso W= 30 y Wc= 11, W>Wc, por lo tanto no se obtienen diferencias significativas entre las heurísticas comparadas.

n=	11
W=	30
Wc=	11

RSVV - FFA					
Instancia	RSVV	FFA	Di	|Di|	Ri
A-n34-k5	0	0.425	-0.425	0.425	4
A-n36-k5	0	1.215	-1.215	1.215	9

A-n39-k5	0	2.088	-2.088	2.088	13
A-n39-k6	0	0.795	-0.795	0.795	7
A-n44-k6	0	0.208	-0.208	0.208	1
A-n45-k6	0.004	0.449	-0.445	0.445	5
A-n45-k7	0	0.314	-0.314	0.314	2
A-n46-k7	0	3.407	-3.407	3.407	18
A-n48-k7	0	2.885	-2.885	2.885	16
A-n53-k7	0.007	1.049	-1.042	1.042	8
A-n54-k7	0.002	6.626	-6.624	6.624	20
A-n55-k9	0	0.346	-0.346	0.346	3
A-n60-k9	0	1.905	-1.905	1.905	12
A-n61-k9	0.001	2.552	-2.551	2.551	14
A-n62-k8	0.018	5.805	-5.787	5.787	19
A-n63-k9	0.002	1.361	-1.359	1.359	11
A-n63-k10	0.001	0.623	-0.622	0.622	6
A-n65-k9	0.004	3.021	-3.017	3.017	17
A-n69-k9	0.006	2.793	-2.787	2.787	15
A-n80-k10	0.004	1.319	-1.315	1.315	10

W+ =	0
W- =	210
W=min(W+,W-)	0
W =	0

n=	20
W=	0
Wc=	52

Para éste segundo caso W= 0 y Wc= 52, W<Wc. Por lo que la Hipótesis nula se rechaza ya que existe diferencias significativas en las heurísticas.

RSVV - CENTROID					
Instancia	RSVV	CENTROID	Di	|Di|	Ri
A-n32-k5	0	12.372	-12.372	12.372	20
A-n33-k5	0	10.136	-10.136	10.136	14
A-n33-k6	0	3.774	-3.774	3.774	3
A-n34-k5	0	4.37	-4.37	4.37	4
A-n36-k5	0	1.877	-1.877	1.877	1
A-n37-k5	0	13.004	-13.004	13.004	22
A-n37-k6	0	8.219	-8.219	8.219	11
A-n38-k5	0	12.192	-12.192	12.192	18
A-n39-k5	0	5.109	-5.109	5.109	5
A-n39-k6	0	6.017	-6.017	6.017	7
A-n44-k6	0	10.672	-10.672	10.672	17
A-n45-k6	0.004	10.169	-10.165	10.165	15
A-n45-k7	0	12.391	-12.391	12.391	21
A-n46-k7	0	8.534	-8.534	8.534	12
A-n48-k7	0	6.71	-6.71	6.71	8
A-n53-k7	0.007	10.594	-10.587	10.587	16
A-n54-k7	0.002	3.599	-3.597	3.597	2
A-n55-k9	0	7.642	-7.642	7.642	10
A-n60-k9	0	5.613	-5.613	5.613	6
A-n61-k9	0.001	16.151	-16.15	16.15	24
A-n62-k8	0.018	14.13	-14.112	14.112	23
A-n63-k9	0.002	9.282	-9.28	9.28	13
A-n63-k10	0.001	6.925	-6.924	6.924	9
A-n64-k9	0.92	13.276	-12.356	12.356	19

W+ =	0
W- =	300
W=min(W+,W-)	0
W =	0

n=	24
W=	0
Wc=	81

Para éste tercer caso W= 0 y Wc= 81, W<Wc. Por lo que la Hipótesis nula se rechaza ya que existe diferencia significativas en las heurísticas.

RSVV - CHFA					
Instancia	RSVV	CHFA	Di	|Di|	Ri
A-n32-k5	0	0.255	-0.255	0.255	23
A-n33-k5	0	0	0	0	8
A-n33-k6	0	0	0	0	8
A-n34-k5	0	0	0	0	8
A-n36-k5	0	0	0	0	8
A-n37-k5	0	0	0	0	8
A-n37-k6	0	0	0	0	8
A-n38-k5	0	0	0	0	8
A-n39-k5	0	0	0	0	8
A-n39-k6	0	0	0	0	8
A-n44-k6	0	0	0	0	8
A-n45-k6	0.004	0	0.004	0.004	17
A-n45-k7	0	0	0	0	8
A-n46-k7	0	0	0	0	8
A-n48-k7	0	0	0	0	8
A-n53-k7	0.007	0	0.007	0.007	18
A-n54-k7	0.002	0	0.002	0.002	16
A-n55-k9	0	0	0	0	8
A-n60-k9	0	0	0	0	8
A-n61-k9	0.001	0.097	-0.096	0.096	20
A-n62-k8	0.018	0.466	-0.448	0.448	24
A-n63-k10	0.001	0.076	-0.075	0.075	19
A-n64-k9	0.92	0.714	0.206	0.206	21
A-n65-k9	0.004	0.256	-0.252	0.252	22

W+ =	171
W- =	221
W=min(W+,W-)	171
W =	171

n=	24
W=	171
Wc=	81

Para éste CUARTO caso W= 171 y Wc= 81, W>Wc por lo tanto no se obtienen diferencias significativas entre las heurísticas comparadas.

RSVV - SWEEP					
Instancia	RSVV	SWEEP	Di	|Di|	Ri
A-n32-k5	0	11.224	-11.224	11.224	10
A-n33-k5	0	19.213	-19.213	19.213	22
A-n33-k6	0	11.725	-11.725	11.725	11
A-n34-k5	0	9.512	-9.512	9.512	2
A-n36-k5	0	10.638	-10.638	10.638	6
A-n37-k5	0	9.716	-9.716	9.716	3
A-n37-k6	0	10.643	-10.643	10.643	7
A-n38-k5	0	19.726	-19.726	19.726	23
A-n39-k5	0	18.127	-18.127	18.127	21

A-n39-k6	0	16.245	-16.245	16.245	18
A-n44-k6	0	16.542	-16.542	16.542	20
A-n45-k6	0.004	10.487	-10.483	10.483	4
A-n45-k7	0	11.78	-11.78	11.78	12
A-n46-k7	0	10.832	-10.832	10.832	8
A-n48-k7	0	6.524	-6.524	6.524	1
A-n53-k7	0.007	10.495	-10.488	10.488	5
A-n54-k7	0.002	13.111	-13.109	13.109	15
A-n55-k9	0	11.09	-11.09	11.09	9
A-n60-k9	0	16.248	-16.248	16.248	19
A-n61-k9	0.001	14.507	-14.506	14.506	16
A-n62-k8	0.018	21.04	-21.022	21.022	24
A-n63-k9	0.002	12.809	-12.807	12.807	13
A-n63-k10	0.001	15.906	-15.905	15.905	17
A-n64-k9	0.92	13.99	-13.07	13.07	14

W+ =	0
W- =	300
W=min(W+,W-)	0
W =	0

n=	24
W=	0
Wc=	81

Para éste último caso W= 0 y Wc= 81, W<Wc. Por lo que la Hipótesis nula se rechaza ya que existe diferencia significativa en las heurísticas.

Como se mostró anteriormente en 2 de los casos la hipótesis nula no se rechaza, sin embargo la distribución de los datos, la normalidad, las gráficas de cajas y bigotes y los mismos resultados nos muestran una diferencia evidente en los algoritmos, en todos los casos obteniendo mejores resultados el algoritmo propuesto RSVV.

En el caso de comparación entre el algoritmo propuesto RSVV con SA, se entiende que por ser los dos un algoritmo de recocido simulado el comportamiento de los datos es parecido pero no así en los resultados. Comparando el algoritmo propuesto RSVV con el CHFA se entiende que el comportamiento de los datos es parecido debido a que ambos en la mayoría de los datos obtienen el óptimo reportado en la literatura, por lo mismo, no existe gran diferencia en cuánto al comportamiento de datos.

Capítulo 7. Conclusiones y trabajos futuros

7.1 Conclusiones

Se presentó un algoritmo heurístico de Recocido Simulado para resolver el Problema de Enrutamiento de Vehículos con Capacidades Homogéneas, el cual consiste de una solución inicial aleatoria factible, la cual es mejorada a continuación por la implementación del método heurístico, se implementaron 3 tipos distintos de vecindario para ampliar la búsqueda del algoritmo durante las perturbaciones de la solución inicial.

Para la evaluación el algoritmo se resolvieron 50 instancias de diferente número de clientes (distintos tamaños), instancias pertenecientes a dos grupos distintos del benchmark de Augeralt et. al., específicamente del grupo "A" y "B", los resultados obtenidos con nuestro algoritmo fueron comparados con los óptimos conocidos de dichas instancias.

El algoritmo dado en esta tesis fue capaz resolver las instancias con las que se evaluó, alcanzando el óptimo conocido en la mayoría de los casos. Del total de las 50 instancias se logró alcanzar el resultado óptimo en 34 de ellas, lo que corresponde a un 68% en el total de las instancias. La calidad de las soluciones que no alcanzaron el óptimo tienen una diferencia menor al 1.8% lo que nos muestra que los resultados quedaron a muy pocas unidades de obtener el óptimo en el resto de las instancias.

7.2 Trabajos futuros

A continuación, se muestra una lista de algunas propuestas para trabajos futuros que no pudieron realizarse en esta tesis por cuestiones de tiempo:

- Agregar nuevos vecindarios al presente trabajo con el fin de alcanzar los resultados óptimos de las instancias faltantes.
- Probar el rendimiento de nuestro algoritmo con instancias de otros grupos del Benchmark de Augeralt et. al. Con el fin de seguir probando su rendimiento.
- Probar el rendimiento del algoritmo con instancias pertenecientes a Benchmark distintos, para verificar su eficacia con distribuciones distintas de clientes o nodos.
- Experimentar con instancias con mayor número de clientes, se proponen más de 100 nodos.
- Probar un algoritmo de Recocido Simulado en una variante distinta del problema, se propone la variante de ventanas de tiempo o multidepósito.

Referencias

[Akhtar, et al., 2017] Mahmuda Akhtar, M.A. Hannan, R.A. Begum, Hassan Basri, Edgar Scavino, *Backtracking search algorithm in CVRP models for efficient solid waste collection and route optimization*, Waste Management, Volume 61, 2017,https://doi.org/10.1016/j.wasman.2017.01.022.

[Alssager et al., 2020] Alssager, M., Othman, Z.A., Ayob, M., Mohemad, R., & Yuliansyah, H. (2020). *Hybrid Cuckoo Search for the Capacitated Vehicle Routing Problem.* Symmetry, 12, 2088.

[Alssager et al., 2020] Alssager, M., Othman, Z.A., Ayob, M., Mohemad, R., & Yuliansyah, H. (2020). Hybrid Cuckoo Search for the Capacitated Vehicle Routing Problem. *Symmetry*, *12*, 2088.

[Altabeeb et al., 2019] Altabeeb, A.M., Mohsen, A.M., & Ghallab, A. (2019). *An improved hybrid firefly algorithm for capacitated vehicle routing problem*. Appl. Soft Comput., 84.

[Asma et al., 2021] Asma M. Altabeeb, Abdulqader M. Mohsen, Laith Abualigah, Abdullatif Ghallab, *Solving capacitated vehicle routing problem using cooperative firefly algorithm*, pplied Soft Computing, Volume 108,2021, https://doi.org/10.1016/j.asoc.2021.107403.

[Azi N., 2010] Nabila Azi, M. G. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research, 202*(3), 756-763.

[Baker y Ayechew, 2003] Baker, B.M., & Ayechew, M.A. (2003). *A genetic algorithm for the vehicle routing problem*. Comput. Oper. Res., 30, 787-800.

[Balinski y Quandt, 1964] Balinski, M. L., & Quandt, R. E. (1964). On an Integer Program for a Delivery Problem. *Operations Research*, *12*(2), 300–304. http://www.jstor.org/stable/167930

[Bell et al., 2004] Bell, JE and McMullen, PR (2004) *Ant colony optimization techniques for the vehicle routing problem*. Advanced Computer Engineering, 18, 41-48. https://doi.org/10.1016/j.aei.2004.07.001.

[Berger y Barkaoui, 2003] Berger, J. and Barkaoui, M., 2003. A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 54(12), pp.1254-1262.

[Clarke, y Wright, 1964] Clarke, G. and Wright, J.R. (1964) *Scheduling of Vehicle Routing Problem from a Central Depot to a Number of Delivery Points*. Operations Research, 12, 568-581.

[Colorni et al., 1991] A. Colorni, M. Dorigo, V. Maniezzo. "Distributed Optimization by Ant Colonies". First European Conference on Artificial Life, 134-142, 1991.

[Dantzig y Ramser, 1959] Dantzig, G. and Ramser, J. (1959*) The Truck Dispatching Problem*. Management Science, 6, 80-91.

[Fukasawa, et al., 2006] Fukasawa, R., Longo, H., Lysgaard, J. et al. *Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem.* Math. Program. **106,** 491–511 (2006). https://doi.org/10.1007/s10107-005-0644-x

[Garey y Johnson, 1979] M. R. Garey and D. S. Johnson. Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA, 1979.

[Garey y Johnson, 2003] Garey, M. & R., Johnson, D.S.: Computers and intractability, A Guide to the theory of NPCompleteness. W.H.Freeman and Company, New York. USA.ed. (2003)

[Gendreau et al., 1994] Gendreau, M., Hertz, A. and Laporte, G., 1994. *A Tabu Search Heuristic for the Vehicle Routing Problem*. Management Science, 40(10), pp.1276-1290.

[Gillett y Miller, 1974] Gillett, B. E., & Miller, L. R. (1974). A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations Research*, *22*(2), 340–349. http://www.jstor.org/stable/169591

[González y González, 2006] González-V, G., & González-A, F. (2006). *Metaheurísticas Aplicadas al ruteo de vehículos. Un caso de estudio. Parte 1: Formulación del problema.* Ingeniería e Investigación, 149-156.

[Granada y Santa, 2016] Granada, M.& Santa, J.. (2016). OPTIMIZACIÓN COMBINATORIA De la teoría a la práctica. Pereira, Risaralda, Colombia: Copyright.

[Hopcroft et al., 2007] Hopcroft J., Motwani R. & Ullman J.. (2007). Teoría de autómatas, lenguajes y computación. Madrid: PEARSON EDUCACIÓN S.A..

[Hosseinabadi et al., 2017] Hosseinabadi, A., Rostami, N., Kardgar, M., Mirkamali, S. and Abraham, A., 2017. *A new efficient approach for solving the capacitated Vehicle Routing Problem*

using the Gravitational Emulation Local Search Algorithm. Applied Mathematical Modelling, 49, pp.663-679.

[Ílhan, 2021] İlhan İLHAN, *An improved simulated annealing algorithm with crossover operator for capacitated vehicle routing problem*, Swarm and Evolutionary Computation,Volume 64, 2021, https://doi.org/10.1016/j.swevo.2021.100911.

[Kirkpatrick et al., 1983] Kirkpatrick , S., Gelatt, C., & Vecchi, M. (1983). Optimization by Simulated annealing. Science, 220(4598), 671-680.

[Laporte G., 1992] Gilbert Laporte, The vehicle routing problem: An overview of exact and approximate algorithms,

[Lin et al.,2009] Shih-Wei Lin, Zne-Jung Lee, Kuo-Ching Ying, Chou-Yuan Lee, Applying hybrid meta-heuristics for capacitated vehicle routing problem, Expert Systems with Applications, Volume 36, Issue 2, Part 1, 2009, Pages 1505-1512, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2007.11.060.

[Mazzeo y Louiseau, 2004] Mazzeo, S. and Loiseau, I., 2004. *An Ant Colony Algorithm for the Capacitated Vehicle Routing*. Electronic Notes in Discrete Mathematics, 18, pp.181-186.

[Mohammed et al., 2012] M. A. Mohammed, M. S. Ahmad and S. A. Mostafa, *"Using Genetic Algorithm in implementing Capacitated Vehicle Routing Problem"*, 2012 International Conference on Computer & Information Science (ICCIS), 2012, pp. 257-262, doi: 10.1109/ICCISci.2012.6297250.

[Mole y Jameson, 1976] Mole, R. H., & Jameson, S. R. (1976). A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion. *Operational Research Quarterly (1970-1977)*, *27*(2), 503–511. https://doi.org/10.2307/3008819

[Morales et al., 2018] Caballero-Morales, SO., Martínez-Flores, JL., Sánchez-Partida, D. (2018). *An Evolutive Tabu-Search Metaheuristic Approach for the Capacitated Vehicle Routing Problem*. In: García-Alcaraz, J., Alor-Hernández, G., Maldonado-Macías, A., Sánchez-Ramírez, C. (eds) New Perspectives on Applied Industrial Tools and Techniques. Management and Industrial Engineering. Springer, Cham. https://doi.org/10.1007/978-3-319-56871-3_23

[Obaid, 2018] Ibrahim Obaid, O. (2018). *Solving Capacitated Vehicle Routing Problem (CVRP) Using Tabu Search Algorithm (TSA)*. Ibn AL-Haitham Journal For Pure and Applied Sciences, 31(2), 199–209. https://doi.org/10.30526/31.2.1949

[Ochelska et al., 2021] Ochelska-Mierzejewska, J.; Poniszewska-Mara´ nda, A.; Mara´ nda, W. *Selected Genetic Algorithms for Vehicle Routing Problem Solving*. Electronics **2021**, 10, 3147. https://doi.org/10.3390/electronics10243147

[Parra & Chavez, 1998] Parra, O. D., & Chavez, M. C. (1998). El problema del transporte. *Centro de Investigación en Ingeniería y Ciencias Aplicadas, Cuernavaca. Morelos./Papadimitriou, CH, Steiglitz, K. Combinatorial*.

[Peralta, 2017] Peralta Abarca, M., 2017. *Algoritmo Distribuido De Recocido Simulado Para El Modelo Del Transporte Vehicular Con Capacidades Homogéneas*. Doctor en Ingeniería y ciencias aplicadas con opción terminal en tecnología eléctrica. Universidad Autónoma del estado de Morelos

[Restrepo y Medina**,** 2008**]** Herrera Restrepo, J., & Medina V., P. (2008). Un problema logístico de programación de vehículos con capacidad finita. *Scientia Et Technica*, *1*(38).

[Shih et al., 2009] Shih-Wei Lin, Zne-Jung Lee, Kuo-Ching Ying, Chou-Yuan Lee, *Applying hybrid meta-heuristics for capacitated vehicle routing problem*, Expert Systems with Applications, Volume 36, Issue 2, Part 1, 2009, https://doi.org/10.1016/j.eswa.2007.11.060.

[Shin y Han, 2011] Shin, K., & Han, S. (2011). A Centroid-based Heuristic Algorithm for the Capacitated Vehicle Routing Problem. Comput. Informatics, 30, 721-732.

[Siarry P., 2016] Siarry, P.. (2016). Metaheuristics. Créteil, France: Springer International Publishing Switzerland.

[Szeto et al., 2011] Szeto, W.Y. & Wu, Yongzhong & Ho, Sin C., 2011. *"An artificial bee colony algorithm for the capacitated vehicle routing problem,"* European Journal of Operational Research, Elsevier, vol. 215(1), pages 126-135, November.

[Taha H., 2012] Taha H.. (2012). Investigación de operaciones. México: PEARSON EDUCACIÓN.

[Tarantilis et al., 2002] CD Tarantilis, CT Kiranoudis y VS Vassiliadis (2002) *A List Based Threshold Accepting Algorithm for the Capacitated Vehicle Routing Problem*, International Journal of Computer Mathematics, 79:5, 537 553, DOI: 10.1080/00207160210948.

[Van Breedam, 1995] Van Breedam, A., 1995. *Improvement heuristics for the Vehicle Routing Problem based on simulated annealing*. European Journal of Operational Research, 86(3), pp.480-490.

[Vigo y Toth,2014] Vigo, D., & Toth, P. (2014). *Vehicle Routing Problems, Methods and Applications.* Philadelphia, USA: MOS-SIAM (Society for Industrial and Applied Mathematics) Series on Optimization.

[Yang y Deb, 2009] Yang, X.-S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.

[Yesodha y Amudha, 2019] R. Yesodha and T. Amudha, *"An Improved Firefly Algorithm for Capacitated Vehicle Routing Optimization,"* 2019 Amity International Conference on Artificial Intelligence (AICAI), 2019, pp. 163-169, doi: 10.1109/AICAI.2019.8701269.

[Yu et al.,2009] Yu, Bin & Yang, Zhong-Zhen & Yao, Baozhen, 2009. "An improved ant colony optimization for vehicle routing problem," European Journal of Operational Research, Elsevier, vol. 196(1), pages 171-176, July.

[Yurtkuran y Emel, 2010] Alkın Yurtkuran, Erdal Emel, *A new Hybrid Electromagnetism-like Algorithm for capacitated vehicle routing problems*, Expert Systems with Applications, Volume 37, Issue 4, 2010, Pages 3427-3433, https://doi.org/10.1016/j.eswa.2009.10.005.

European Journal of Operational Research, Volume 59, Issue 3, 1992, Pages 345-358, ISSN 0377-2217, https://doi.org/10.1016/0377-2217(92)90192-C.

Cuernavaca, Morelos a 24 de marzo del 2023.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
ENCARGADO DE DESPACHO DE LA DIRECCIÓN DE LA FACULTAD
DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA DE LA UAEM.
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Dalia Vanessa Arce Ortega, con matrícula 10036986, con el título Implementación de un algoritmo heurístico de Recocido Simulado para el Problema de Enrutamiento de Vehículos con Capacidades Homogéneas por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. Federico Alonso Pecina
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

FEDERICO ALONSO PECINA | Fecha:2023-03-25 23:19:47 | Firmante

V4vQmG7ojHSh6fOWit+kyzK2zFs5BmXGzwBxMHiiTE0IE4l2etl9xECv41DZGSH+2q9DBeo5FO1zccDOmUNofgcbifLKA8LC2qz0wDGbV8edjjPgRroqkgyLrFOFvYWL7HdkEOBh9t603jVRHRdS7RApZlfzHiwl4O6Bsw+/SfeHscZpNj3TCeYS/fsBbDauP/MZB46NmTvSi2KwTl7gvPDInOxTKSh8ZTzRvnp3g2Uqc5DuV8mjjidMj2XlSEAgP2CIKVNh4A+yGKWSq+3Ex8gyF7TKml+rlqlBNik75vkc7jNiZnKGqcU8d5IDWINXlTfd2362iztZcix7cH/0OQ==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

UfaGygJHw

https://efirma.uaem.mx/noRepudio/NdsO9CzYsNXPfW4HGRGhPpiwnMaTHnr9

Cuernavaca, Morelos a 24 de marzo del 2023.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
ENCARGADO DE DESPACHO DE LA DIRECCIÓN DE LA FACULTAD
DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA DE LA UAEM.
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Dalia Vanessa Arce Ortega, con matrícula 10036986, con el título Implementación de un algoritmo heurístico de Recocido Simulado para el Problema de Enrutamiento de Vehículos con Capacidades Homogéneas por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. José Crispín Zavala Díaz
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

JOSE CRISPIN ZAVALA DIAZ | Fecha:2023-03-25 16:46:24 | Firmante

HPuMZqGUsjIGGM81lEzitlb6J5oBLEPV54sbuwBTp/RHljMEQ4m7wtSYMzBY+iMBTkyafBO5u2aep6XZql9SXcv18ZneoRojXV+druB3tJRcLg/W+uT3gjhPRN/q2h+U0oPDv5xGZ
AbXnH9ixCqkjJITAtjgufND6GtV550cgp3f/Yfi/5tYfjxuOswA/xQbfmU4GFHJvbpJW4N/tHXxFs45+V6seJhIm1fOUgeSsanQNQQ8OZbMU6WOMIpPmS3GQfHxEkAQk8pb11lLYAW
A+MqwKMIkQMBwmjoFB2ZaTx9VDZs7HRyZ7AmbQAwew+cLd6xY86PSTIxe5+FWUEiprw==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

hHJqVR4Pd

https://efirma.uaem.mx/noRepudio/ehFwmt8a27Seup77OSMBy5JAiGsqrCfI

Cuernavaca, Morelos a 24 de marzo del 2023.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
ENCARGADO DE DESPACHO DE LA DIRECCIÓN DE LA FACULTAD
DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA DE LA UAEM.
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Dalia Vanessa Arce Ortega, con matrícula 10036986, con el título Implementación de un algoritmo heurístico de Recocido Simulado para el Problema de Enrutamiento de Vehículos con Capacidades Homogéneas por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. Marco Antonio Cruz Chávez
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

MARCO ANTONIO CRUZ CHAVEZ | Fecha:2023-03-31 19:11:28 | Firmante

bEc3dICieA+m085NXP5X4iLpO7mSMcfotEday2SFstYuvNLm0Ubrn7cMKqRe/GJDftxSO9ePXOlR4qjsjHWU86iQ310PGJqNsP3f12y5mxcFEK4gxn2r/1Dxo2yUpbKETkoZ6MXHvicYjlYiTwL0CXQnjBUv5ZbdQ/G+r5ETjGmm8LKQV6VXUbtr8lFvr+4SclUGHtrOlk9j2ylaHFS9fiF2ENsQmHqCZEd2yEydCLUO8JDy+kcIo51gIGo7E6/5hz/n25ZAEPR46rIuMKZ3iFpunXCIsmSdCzJ+qrJWIir+RntVnKGqr1re5sqNtGUUmQ+x/TzuloX1bs9RkY+zcQ==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

Sa0QFLkYR

https://efirma.uaem.mx/noRepudio/YxtcsJuqhRRB031ijf2eq2XT9feXMXNi

Cuernavaca, Morelos a 24 de marzo del 2023.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
ENCARGADO DE DESPACHO DE LA DIRECCIÓN DE LA FACULTAD
DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA DE LA UAEM.
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Dalia Vanessa Arce Ortega, con matrícula 10036986, con el título Implementación de un algoritmo heurístico de Recocido Simulado para el Problema de Enrutamiento de Vehículos con Capacidades Homogéneas por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dra. Irma Yazmín Hernández Báez
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

IRMA YAZMIN HERNANDEZ BAEZ | Fecha:2023-03-27 10:21:04 | Firmante

Yxn4RBGk7dAaq7eHOvMLrxFXiWwsBaC0haN7B/OyHH0u3LWn88kt47e4WzrdHyA6AnjJypZC1eKkJE7HSNCqfqdp4gh10p9uJ3N8v8GauM9k4nJhmHrY3LvLROptlfAypVxLftLFXSHqTxNcg/c893yuq3nWZhiq/oibU/ZCKtQBBXwXQinRL0UoBKZjKPNRlBFSBvkazRdSKT37gI/raPvOloXeBhKM76r0MjejJ9Non5jwC60fRNVjLiZA35MbxyC7A/LKWCZLmNthzUdPAvymxhB5gTOXiDxg4vv4BJ8N1RArDLbmEDXSC+3EoGHAiEV7QByAiQJOxqVEsHVodg==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

8Y3kZH7WT

https://efirma.uaem.mx/noRepudio/coo4wTgUNat78tfU67jujV0wlv5ingOa

Cuernavaca, Morelos a 24 de marzo del 2023.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
ENCARGADO DE DESPACHO DE LA DIRECCIÓN DE LA FACULTAD
DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA DE LA UAEM.
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Dalia Vanessa Arce Ortega, con matrícula 10036986, con el título Implementación de un algoritmo heurístico de Recocido Simulado para el Problema de Enrutamiento de Vehículos con Capacidades Homogéneas por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

M.O.C.A. Víctor Hugo Pacheco Valencia
Profesor Externo
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

VICTOR HUGO PACHECO VALENCIA | Fecha:2023-03-25 09:13:35 | Firmante

N5oER/5VCi0Tau4G1yRNAhIf+orkRvDFGYkb+GaFfLjIQf6fGElhoIAob9fbQ3FIlq7WoFHzP7nYL1L2U5KDQdyLNg7ZL1Dbgtc+qkCbo5P8G7eUmOD9IzstqeGt2NQfgybgIMBoGHT9DSDzV0UNNQmjql5n2fp26y9MrCH5BowPNkBCyql4L9xpCLJmog19uYhcAtmG5HZEhQpSEBSVZu+nrJoNDDFxPMLNYAKiywFLiMzaO4yS6QWKeMQWIf1DsfaZi/gHGv5Rh2H+SsxtLnJFqhxdgAQXNlFp2Syipw+Ar0JVZkFZoBGAV7xZSpRaiAeMZrqAOI0Mn9YYghvrpg==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

Jht7RuGBZ

https://efirma.uaem.mx/noRepudio/AGrSYt3k6ZQZYjLJs12G4fMtH9A0gxSi