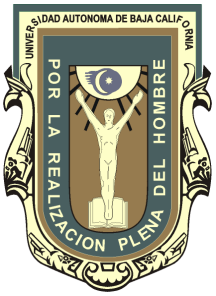
Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño



Maestría y Doctorado en Ciencias e Ingeniería

Algoritmo metaheurístico GRASP aplicado al problema de rutas de vehículos con ventanas de tiempo

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN INGENIERÍA

Presenta

ALMA DANISA ROMERO OCAÑO

Ensenada, Baja California, Julio del 2017.

Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Algoritmo metaheurístico GRASP aplicado al problema de rutas de vehículos con ventanas de tiempo

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN INGENIERÍA

Presenta

Alma Danisa Romero Ocaño

Aprobada por:

Dra. María de los Ángeles Cosío León
Director de Tesis

Dra. Anabel Martínez Vargas
Miembro del Comité

Dr. Álvaro Alberto López Lambraño
Miembro del Comité

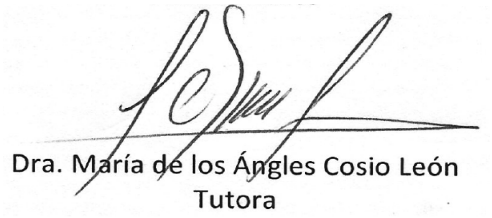
Dr. Miguel Enrique Martínez Rosas
Miembro del Comité

Ensenada, Baja California, Julio del 2017.

Resumen del protocolo de **Alma Danisa Romero Ocaño**, presentada como requisito parcial para ingresar en le posgrado de DOCTORADO EN CIENCIAS del programa de Maestría y Doctorado en Ciencias e Ingeniería (MYDCI) de la UABC. Ensenada Baja California, México, mayo del 2017.

Metaheurísticas multi-objetivo para optimizar el problema de rutas de vehículos eléctricos con flota heterogénea, restricciones de tiempo y estaciones de recarga

Resumen Aprobado por:



Los Vehículos eléctricos (VEs) a menudo se han sugerido como una solución útil para reducir el consumo de petróleo y las emisiones de contaminantes atmosféricas. Debido a la eficiencia energética y las ventajas ambientales sobre los vehículos convencionales, el futuro de los vehículos eléctricos parece prometedor, sin embargo, la integración de los VEs en sistemas de energía eléctrica plantea nuevas técnicas económicas, políticas y desafíos regulatorios.
En este trabajo se aborda la optimización del problema de rutas de vehículos eléctricos con restricciones de tiempo, flota heterogénea y estaciones de recarga con el diseño de algoritmos multiobjetivos basados en las metaheurísticas MOEA/D y NSGA-II, buscando aproximarnos a soluciones de buena calidad, comparando los resultados con los reportados en la literatura.

Palabras Clave: *multi-objetivo*, *vehículos eléctricos*, *E-VRPTW*.

Dedicatoria

A Dios por sus bendiciones durante mi vida, por permitirme cumplir con mis objetivos, por brindarme salud, bienestar y felicidad.

A mis padres por el esfuerzo que siempre hicieron durante su vida por que lograra mis objetivos y sueños; segura estoy que algún día volveremos a vernos.

A mi esposo ***Víctor Manuel*** *por ser mi compañero de vida y compartir conmigo esta y todas las aventuras que aún nos faltan por vivir; por su apoyo, paciencia, consejos y fortaleza en el logro de éste trabajo. Te amo y te admiro.*

A mis hijas, ***Danissa*** *y* ***Arlenne****, mis dos princesas que me hacen sentir orgullosa de ser su madre y a quienes agradezco su paciencia y su apoyo, han sido mi motivación más grande para concluir con éxito esta tesis, un profundo Gracias por no reclamar el tiempo que les he robado por estar en esta labor que tanto me apasiona, por aceptar los cambios y asimilarlos como retos. Su felicidad siempre será la mía, las amo por siempre.*

A mis hermanos ***Icela, Oliva y Carlos****, les agradezco no solo por su apoyo incondicional, si no también por existir en mi vida, por esos momentos de tristezas y felicidad que hemos compartido durante nuestras vidas, gracias por esos hermosos sobrinos que tanto quiero y extraño. Se siente extraño no nombrarte, siempre te recuerdo hermano* ***Javier****.*

A mis suegros ***Concepción y Paula****, les agradezco su apoyo, su amor y muestras de cariño para mis hijas y mi persona.*

Agradecimientos

Quiero expresar mi agradecimiento:

A mi directora de tesis ***Dra. María de los Ángeles Cosío León****, sus conocimientos, su paciencia, su orientación, su manera de trabajar y motivación, han sido fundamentales para la culminación de esta tesis.*

A mis sinodales ***Dr. Miguel Enrique Martínez Rosas y Dr. Álvaro Alberto López Lambraño*** *por sus consejos, criticas y apoyo durante el desarrollo de esta tesis. A la* ***Dra. Anabel Martínez Vargas****, que con su colaboración, su aporte y participación activa, a enriquecido el presente trabajo.*

A los docentes que durante mi estancia en la Universidad, compartieron sus conocimientos y experiencias.

A mis compañeros y amigos, por los buenos momentos que compartimos. Todos hemos crecido profesional y personalmente, éxito siempre.

Al Tecnológico Nacional de México, que a través del Instituto Tecnológico de Agua Prieta (ITAP), me brindó el apoyo y la oportunidad de realizar los estudios de éste posgrado. Resaltando la oportunidad, el soporte y motivación que siempre brindó a mi persona el ***Dr. José Víctor García Castellanos****, director del ITAP. Agradezco a la* ***M.C. Silvia P. Gutiérrez****, por su ayuda como siempre, finalmente no puedo dejar de agradecer a la* ***M.C. Lucía Castro León*** *por su apoyo y palabras de aliento, pero sobre todo sus muestras de cariño y sincera amistad.*

Al personal de la Universidad Autónoma de Baja California (UABC) de la Facultad de Ingeniería y Diseño, por sus atenciones, amabilidad y disposición para ayudarme.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado y a la Facultad de Ingeniería y Diseño de la Universidad Autónoma de Baja California (UABC) por las facilidades otorgadas para la realización de éste trabajo.

Índice general

1. Introducción **1**
- 1.1. Planteamiento del problema . . . 2
- 1.2. Justificación . . . 3
- 1.3. Objetivo general . . . 4
- 1.4. Objetivos específicos . . . 4
- 1.5. Secuencia de la tesis . . . 4

2. Revisión de Literatura **6**
- 2.1. Antecedentes . . . 6
- 2.2. Problema de ruta de vehículos con ventanas de tiempo ($VRPTW$) . . . 12
  - 2.2.1. Modelo matemático . . . 13
  - 2.2.2. Revisión de trabajos en la literatura del VRPTW . . . 15
- 2.3. Métodos de solución . . . 18
  - 2.3.1. Algoritmos exactos . . . 19
  - 2.3.2. Métodos heurísticos. . . . 20
  - 2.3.3. Métodos metaheurísticos . . . 22
- 2.4. $GRASP$ aplicado al $VRPTW$ . . . 25

3. Desarrollo de Investigación **28**
- 3.1. Metodología DIMMA . . . 28
- 3.2. Algoritmo $GRASP$ aplicado al $VRPTW$ . . . 32
  - 3.2.1. Instancias . . . 33
  - 3.2.2. Fase de construcción . . . 37
  - 3.2.3. Fase de mejora - búsqueda local . . . 41

4. Pruebas y Resultados **46**
- 4.1. Resultados del $G\text{-}VRPTW$ en etapa de construcción, Fase 1 . . . 46
- 4.2. Resultados con búsqueda local en la Fase 1 . . . 50
- 4.3. Diseño de experimentos . . . 55
  - 4.3.1. Aplicación del método Taguchi . . . 56
  - 4.3.2. Resultados finales . . . 65

5. Discusiones y Conclusiones **67**
5.1. Discusiones 67
5.2. Conclusiones 69
5.3. Trabajo futuro 70
5.4. Aportaciones académicas 71

Índice de Figuras

1.1. Problema de rutas de vehículos con ventanas de tiempo ($VRPTW$) . . 3

2.1. Problema del agente viajero (TSP) 7
2.2. Problema TSP y VRP 8
2.3. Variantes del problema de ruta de vehículos 10
2.4. Poblema de rutas de vehículos 13
2.5. Técnicas de optimización 19
2.6. Búsqueda local en vecindarios 22

3.1. Evaluación del desempeño de la calidad de las soluciones en un problema de minimización 30
3.2. Pasos en cada fase de la metodología DIMMA 31
3.3. Instancia de Solomon 35
3.4. Diagrama de flujo de la metaheurística $GRASP$ 37
3.5. Ejemplo de ordenamiento No. 1 38
3.6. Ejemplo de ordenamiento No. 2 38
3.7. Ejemplo de ordenamiento No. 3 38
3.8. Construcción de una ruta 41
3.9. Destrucción de ruta con vértice único 42
3.10. Reubicación de vértice más cercano del depósito 43
3.11. Destrucción de una ruta 43
3.12. Aplicar TSP a una ruta 44

4.1. Resultados en fase de construcción de 25 clientes 48
4.2. Resultados en fase de construcción de 50 clientes 49
4.3. Resultados en fase de construcción de 100 clientes 49
4.4. Resultados en fase de mejora de 25 clientes 52
4.5. Resultados en fase de mejora de 50 clientes 52
4.6. Resultados en fase de mejora de 100 clientes 53
4.7. Comparación de resultados en fase de construcción y búsqueda local . . 54
4.8. Resultados por instancia del mejor ordenamiento 55
4.9. Gráfica de evaluación de la normalidad de datos 60
4.10. Gráfica de efecto principal para medias 62
4.11. Gráfica de efecto principal para S/R 63

4.12. Gráfica comparativa de los resultados obtenidos 64

5.1. Gráfica comparativa de los resultados finales obtenidos en las aproximaciones a la resolución del problema 69

Índice de tablas

2.1. Relación de artículos de investigaciones recientes del problema de ruta de vehículos 12
2.2. Aplicaciones del *VRPTW* 18
2.3. Revisión de literatura *GRASP* aplicada a *VRPTW* 27

3.1. Tabla de clasificación de instancias 34

4.1. Resultados obtenidos en la fase construcción 47
4.2. Resultados obtenidos en la fase de búsqueda local 51
4.3. Comparación de los resultados obtenidos en la fase de construcción y fase de búsqueda local 53
4.4. Concentrado de resultados por cada instancia de prueba en **Fase 1** . . 55
4.5. Tabla de factores seleccionados 58
4.6. Arreglo L_9 propuesto por Taguchi 59
4.7. Concentrado de factores y niveles en la aplicación del método Taguchi . 60
4.8. Promedios del % Gap en cada una de las instancias 61
4.9. Resultados de medias y señal a ruido (S/R) 62
4.10. Niveles sugeridos por Taguchi 63
4.11. Promedios del % Gap en el 80 % de las instancias 65
4.12. Promedio de % Gap por tipo de calibración 66

5.1. Comparación con otras investigaciones en instancias de 100 clientes . . 68

Capítulo 1

Introducción

El sistema de transporte para la mayoría de las organizaciones es un proceso crítico, ya que su eficiencia contribuye a mejorar el desempeño de una empresa, cumplir con los requerimientos de los clientes en tiempo y con calidad, adquirir una ventaja competitiva y bajos costos en sus productos.

Actualmente las empresas buscan nuevos mercados y oportunidades de crecimiento. La mejora en los servicios de transporte y distribución es uno de sus principales desafíos cuando se interesan en el diseño adecuado de la cadena de abastecimiento, ya que esto le permitirá reducir costos, cumplir con los requerimientos de sus clientes y evaluar la posibilidad de ampliar mercados [1].

El proceso de mejorar el servicio de transporte en una empresa, implica contar con rutas que minimicen los costos de transporte, por lo que se requiere optimizar este proceso. Uno de los problemas más comunes y estudiados en la optimización de operaciones logísticas, es el problema de ruteo de vehículos (VRP, por sus siglas en inglés) [2]. Estos problemas plantean la búsqueda de la solución óptima tomando en cuenta varias restricciones como lo son: el número de vehículos, capacidad, lugares de destino (clientes) y demanda de los clientes, entre otras.

Al formular este tipo de problemas se puede incluir un amplio número de variables y parámetros. Este tema presenta un interés práctico y académico por ser un problema de optimización combinatoria y pertenecer a la clase NP-difícil, ya que presentan un alto grado de dificultad de resolución, debido a las múltiples interrelaciones de sus variables y al gran tamaño que tienen los datos de entrada, y no es posible resolverlos en tiempo polinomial [3], [4], [5].

En 1956 Flood [6] introdujo el primer problema de este tipo, el del agente viajero o TSP (Travelling Salesman Problem por sus siglas en inglés), llamado así porque se describe en términos de un agente vendedor que debe visitar una cantidad determinada de ciudades en un solo viaje, de tal forma que inicie y termine en el mismo lugar. El agente

deberá determinar cuál será la ruta que seguirá para visitar cada ciudad una sola vez y regresar, recorriendo la mínima distancia. Este problema es considerado básico en los temas de rutas de vehículos, posteriormente se fueron considerando otras variables y restricciones, apegadas a la realidad, generándose diferentes investigaciones y algoritmos de solución.

1.1. Planteamiento del problema

La asignación de rutas de reparto a una flota de vehículos es un problema de decisión muy común en las empresas dedicadas a la distribución de bienes o servicios, ya que todos los días hacen entrega de sus productos o servicios enfrentándose a la necesidad de diseñar rutas que minimicen distancia y por ende el costo de transporte.

Por otra parte los problemas de rutas de vehículos en la realidad presentan un gran número de condiciones que afectan al modelo que lo describe matemáticamente, haciéndolo más difícil de resolver de manera exacta, estas condiciones son por ejemplo la capacidad del vehículo, la cuál puede ser igual para todos los vehículos (homogénea) o no (heterogénea); otra condición puede ser el número de depósitos donde se recoge la carga, la demanda del cliente, el tráfico, la distancia, otro factor importante es el tiempo en el que se debe entregar los productos a los clientes.

En base a los antecedentes anteriores, en el presente trabajo se realizó la evaluación de un algoritmo metaheurístico, considerando las restricciones de capacidad y demanda, además de incluir ventanas de tiempo, es decir, atender a los clientes en el horario requerido por ellos, visitando a los clientes solo una vez estableciendo como función objetivo minimizar la distancia recorrida y con ello el costo de transporte.

Considerando las restricciones anteriores, el algoritmo aplicado en esta investigación dará soluciones a la variante del VRP [2], denominada problema de rutas de vehículos con ventanas de tiempo (*VRPTW*, Vehicle Routing Problem with Time Windows) estudiado por primera vez por Pullen y Web en 1967 [7].

Este tipo de problemas considera las siguientes restricciones: demanda, capacidad, visitar al cliente solo una vez, los vehículos inician y terminan en el depósito y además considera las ventanas de tiempo, es decir el intervalo de tiempo al cuál el cliente está dispuesto a recibir su mercancía. En la Figura 1.1, se puede observar un esquema de tres rutas en las que los vehículos inician y terminan en el depósito, las ventanas de tiempo se representan por cada cliente, el tiempo inicial de la ventana de tiempo, a la que se le llama a y el tiempo final que se denomina como b [8].

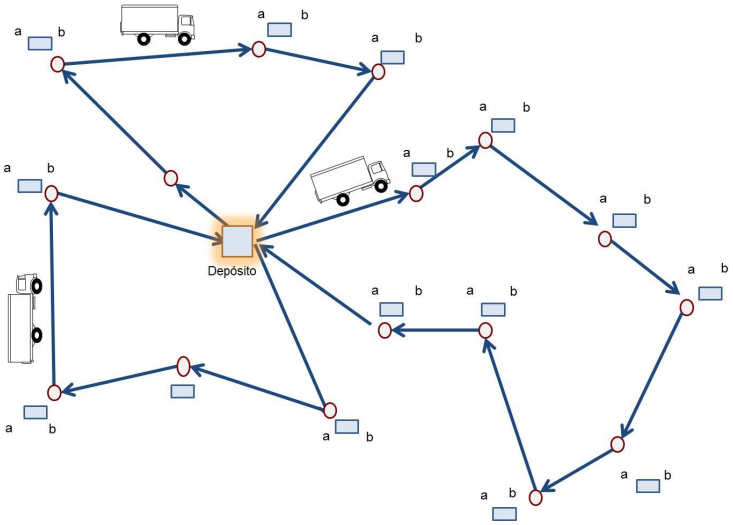


Figura 1.1: Representación de un problema *VRPTW*

1.2. Justificación

En la actualidad, para las empresas que distribuyen bienes o servicios a varios clientes o dependencias, planificar las rutas y satisfacer a sus clientes ha adquirido relevancia debido a los costos de transporte y la competitividad en el mercado.

En muchos de los casos los gastos de producción, almacenamiento, transporte, entre otros, tienden a ser muy elevados, lo que se refleja drásticamente en el precio del producto, por lo que es necesario minimizar dichos costos sin afectar la calidad del producto y satisfacción al cliente.

El problema de distribuir productos desde ciertos depósitos a sus usuario finales juega un papel importante en el área de logística y planifación de una empresa, y al optimizar las rutas de transporte puede significar considerables ahorros en los gastos. Esos potenciales ahorros justifican en gran medida la utilización de técnicas de investigación operativa como facilitadoras de la planificación, dado que se estima que los costos del transporte representan entre el 10 % y el 20 % del costo final de los bienes [5].

Además en la actualidad el nivel de costos de combustible aumenta de manera sostenida en el tiempo, así que generar una metodología que presente ahorros en los costos actuales de una empresa y mejore la estructura de planificación operacional de la misma, da el punto inicial para el desarrollo del presente proyecto.

1.3. Objetivo general

Analizar la calidad de las soluciones que proponga el algoritmo metaheurístico GRASP al problema de rutas de vehículos con ventanas de tiempo.

Con el propósito de alcanzar el objetivo general, se desprenden los siguientes objetivos específicos:

1.4. Objetivos específicos

- *Estudiar el problema de rutas de vehículos con ventanas de tiempo*
- *Analizar el modelado y técnicas empleadas en la literatura, para proponer soluciones al problema de rutas de vehículos, en particular, la variante de ventanas de tiempo con la metaheurística GRASP.*
- *Analizar y ajustar el algoritmo GRASP, para la búsqueda de soluciones en el problema de rutas de vehículos con ventanas de tiempo, al minimizar la longitud de la ruta*
- *Seleccionar y evaluar un método de ordenamiento para GRASP en la etapa de construcción, utilizarlo para proponer soluciones al problema de rutas de vehículos con ventanas de tiempo, minimizando la longitud de la ruta*
- *Calibrar los parámetros del algoritmo GRASP considerando las instancias de Solomon*
- *Comparar los resultados obtenidos una vez calibrado el algoritmo con los mejores resultados conocidos en la literatura de las instancias de Solomon en cuanto a distancia total de la ruta.*

1.5. Secuencia de la tesis

Este trabajo está dividido en los siguientes capítulos:

- Capítulo 2. Se hace una revisión de la literatura relacionada con los problemas de rutas de vehículos, detallando los antecedentes y la descripción del problema de rutas de vehículos con ventanas de tiempo. Se hace una revisión del algoritmo metaheurístico *GRASP* (Greedy Randomized Adaptative Search Procdures), el cual es un procedimiento de búsqueda local voraz, adaptativo y aleatorio.
- Capítulo 3. Se describe la metodología DIMMA (Metodología de diseño e implementación para algoritmos metaheurísticos), base para el desarrollo del presente proyecto. De igual forma, se describen las diferentes etapas que se realizaron en

la investigación. Se explica en que consiste el algoritmo y las búsquedas locales que se modelaron y programaron para obtener los resultados mostrados.

- Capítulo 4. Se presentan los resultados obtenidos en la modelación y programación del algoritmo en C++. Se detallan y analizan los resultados obtenidos antes de aplicar las búsquedas locales, se presentan de igual forma los resultados obtenidos al aplicar el método Taguchi

- Capítulo 5. Finalmente, se presentan las conclusiones basadas en los resultados obtenidos, las observaciones y experiencias. Se plantea el trabajo futuro derivado de esta investigación que contribuya a la solución de problemas de rutas de vehículos.

Capítulo 2

Revisión de Literatura

En este capítulo se hace una revisión de la literatura relacionada con los problemas de rutas de vehículos, detallando los antecedentes y la descripción del problema de rutas de vehículos con ventanas de tiempo. Además, se hace una revisión del algoritmo metaheurístico *GRASP* (Greedy Randomized Adaptative Search Procdures), el cual es un procedimiento de búsqueda local voraz, adaptativo y aleatorio.

2.1. Antecedentes

En 1956 Flood [9] por primera vez estudió el problema de rutas de vehículos, denominado agente viajero o *Travelling Salesman Problem* (TSP), que consiste en un agente vendedor que debe visitar una cantidad determinada de ciudades en un solo viaje, de tal forma que inicie y termine en el mismo lugar. El agente deberá determinar cual será la ruta que seguirá para visitar cada ciudad una sola vez y regresar recorriendo la mínima distancia y al menor costo de transporte.

A partir de esta propuesta inicial se empezaron a considerar diferentes variables, en 1959 Dantzing y Ramser [5]; modelan el despacho de combustible a través de una flota de camiones a diferentes estaciones de servicio, desde una terminal, convirtiéndose este trabajo en la base para un desarrollo posterior de otras formulaciones en las que se incrementan el número de variables y restricciones [5],[9],[10].

En 1960 se encuentra la primera referencia del TSP múltiple (m-TSP) con Miller, Tucker y Zemlin [9]. Este problema es semejante al TSP, se considera un depósito y múltiples vehículos, es decir múltiples agentes viajeros. El objetivo es contar con una ruta por cada agente viajero y una para cada vehículo, cumpliendo las restricciones del TSP, las cuales son:

- a) Cada cliente es visitado una vez por uno de los vehículos.
- b) Cada ruta debe comenzar y finalizar en el depósito y,

- c) En un m-TSP a cada cliente se le asocia una demanda y cada vehículo cuenta con cierta capacidad, razón por la que se concluye que el problema del agente viajero da origen al problema de ruteo [5].

En la Figura 2.1 se representa gráficamente la solución de un TSP, en donde se aprecia que la ruta uno es como actualmente se está realizando el servicio a los clientes, en el número dos se representan todos los posibles caminos que podríamos seguir con el fin de recorrer todas los clientes y brindarles el servicio, en la ruta número 3, se modela la solución óptima que nos proporciona la distancia más corta y con menor costo de transporte.

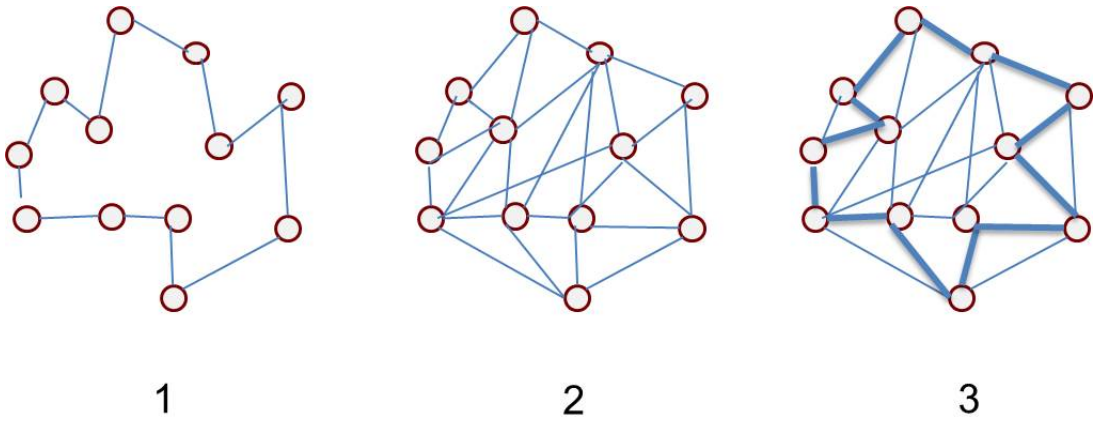


Figura 2.1: Representación de un problema TSP

Por lo anterior se puede observar que el m-TSP, problema de los múltiples agentes viajeros puede ser asumido como un VRP y aún más, como un CVRP (Capacited VRP), problema de ruteo de vehículos donde la capacidad es parte de las restricciones a considerar al momento de realizar la formulación matemática [5]. En la Figura 2.2 a) se muestra un problema TSP, donde el objetivo es visitar a todos los clientes, en una sola ruta, iniciando y terminando en el depósito y en 2.2 b) se aprecian varias rutas, restringidas por la capacidad de los vehículos, mostrando diferentes TSP, es decir, un m-TSP o un VRP.

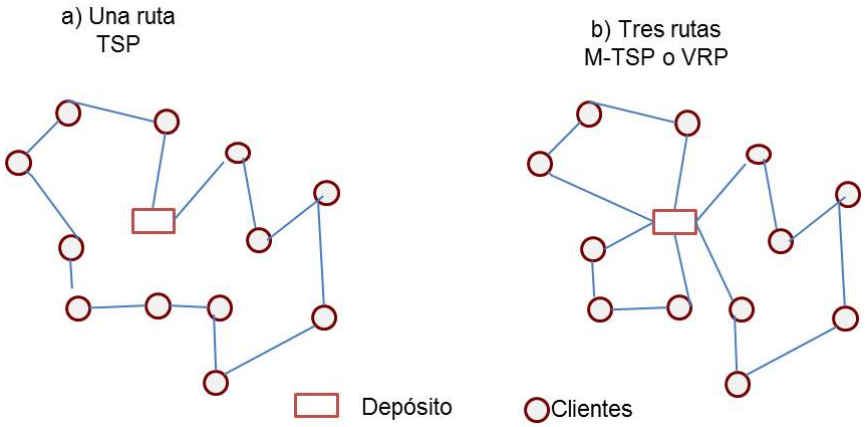


Figura 2.2: Representación gráfica de TSP y VRP(m-TSP)

En función de lo anterior las variables que se deben de considerar para un VRP son:

1. Cantidad de clientes.
2. Localización de los clientes.
3. Centros de distribución.
4. Capacidad de vehículos.
5. Demandas de los clientes.
6. Tiempos y costo de transportación.

A partir del VRP o CVRP generalizados se desprenden dos grandes categorías: a) El VRP homogéneo y b) el VRP heterogéneo. El VRP homogéneo se refiere a características comunes en las que todos los nodos manejan el mismo recurso como distancia, ventanas de tiempo, retornos y entregas fraccionadas. Por su parte, el VRP heterogéneo se refiere a componentes diferentes, en las que cada nodo maneja recursos distintos bien sea flota de vehículos, depósitos, viajes y componentes estocásticos en algunos casos [9]. En la Figura 2.1, se muestran las variantes del VRP, el nombre del modelo, el autor o autores que proponen su formulación matemática y el año de realización.

Los principales VRP homogéneos que se han considerado son:

- DCVRP (Distance CVRP). Es la primer variante del VRP es la que considera la restricción de capacidad (CVRP) y al incluir la restricción de máxima distancia se le llama DCVRP [9].

- *VRPTW*, En esta variante se toma como restricción adicional una ventana de tiempo para cada consumidor, asignando un intervalo de tiempo en el cual el consumidor debe de ser atendido [5].
  En el instante en el que los vehículos salen del centro de distribución, conocido en la literatura como depósito se empieza a considerar el tiempo de recorrido para cada arco y el tiempo de servicio adicional para cada consumidor. El servicio de cada consumidor debe empezar dentro de la ventana de tiempo asociada, en caso que el vehículo llegue antes del tiempo establecido, éste debe esperar hasta el instante de tiempo en el que el servicio debe empezar. Las ventanas de tiempo están definidas de tal manera que todos los vehículos salen del centro de distribución en el tiempo cero [5], [7].
  En este tipo de problemas se debe encontrar una cantidad de recorridos con el mínimo costo de tal manera que cada ruta o recorrido inicia en el centro de distribución y cada centro de consumo es visitado solamente por una ruta.
  La suma de las demandas de los centros de consumo visitados por un recorrido no debe exceder la capacidad del vehículo, para cada centro de consumo el servicio comienza dentro de la ventana de tiempo y el vehículo se detiene por instantes de tiempo. Este tipo de problema tiene extensiones como VRPMTW, VRPTD y OVRPTW [9]. En la Sección 2.2 abordaremos nuevamente esta variante ya que es motivo de estudio de este trabajo.

- VRPB. En este tipo de problemas los consumidores pueden demandar o retornar algunas mercancías. Es necesario tener en cuenta la capacidad del vehículo para el retorno [9].

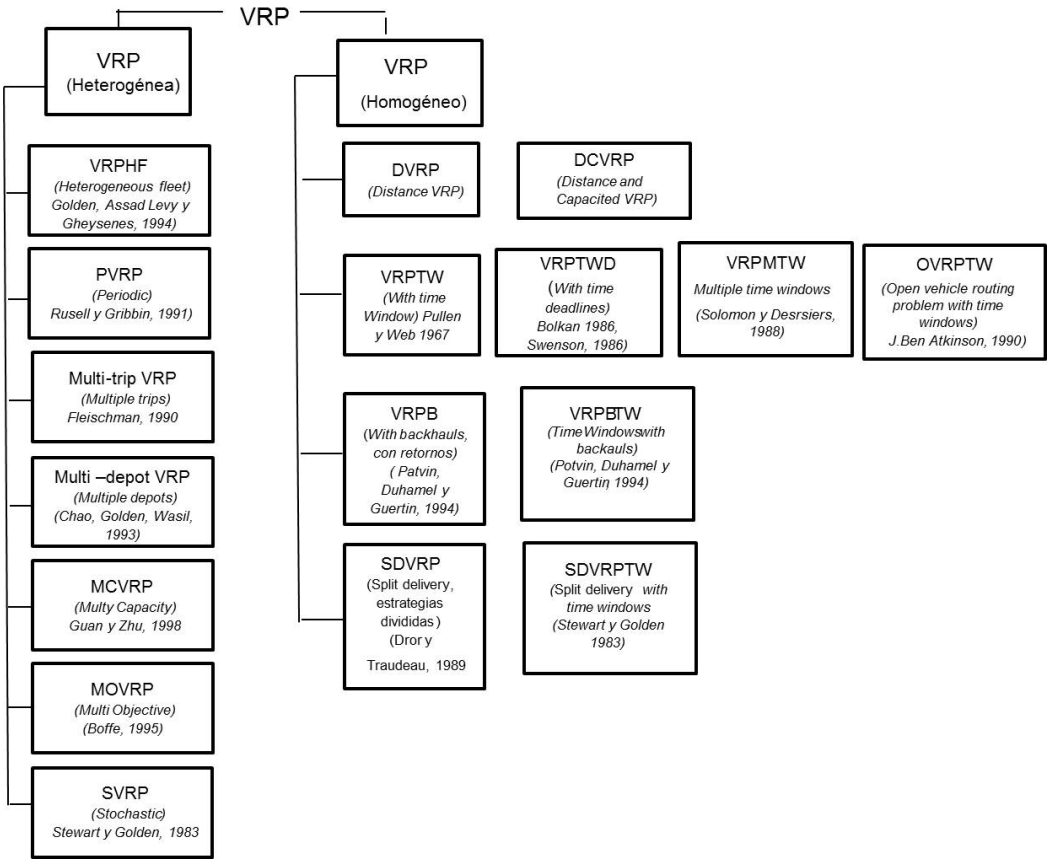


Figura 2.3: Variantes del problema de ruta de vehículos

- SDVRP. Es una variación del VRP en donde se permite que el mismo cliente pueda ser atendido por diferentes vehículos siempre y cuando se reduzca el costo total [9], [5].

Los modelos VRP heterogéneos que son más estudiados son:

- VRPHF: En esta formulación se asume que la cantidad de vehículos de cada tipo es ilimitada, se decide sobre las rutas y la composición de la flota de vehículos a utilizar [9], [5].
- PVRP: En los VRP clásicos, típicamente el periodo de planeación es de un solo día, en el caso del VRP periódico (PVRP) el modelo se extiende a un período de planeación de m días [9], [5].
- Multi-Trip VRP: Consiste en que cada vehículo puede llevar a cabo varias rutas en el mismo periodo de planeación. Resolver este tipo de problema no sólo implica

el diseño de un conjunto de rutas, sino también la asignación de esas rutas a los vehículos disponibles. Esto hace que este tipo de problema sea muy práctico a nivel operativo, en el cual los horarios diarios de los conductores deban estar diseñados para una flota de vehículos fija [9], [5].

- MCVRP: Consiste en transportar más de una cantidad de objetos a la vez, es decir, que los vehículos pueden hacer entregas de diferentes productos a la vez [5].
- MOVRP: Esta variante puede establecer diferentes funciones objetivos como pueden ser: costo, beneficio, vehículos, entre otros [9].
- SVRP: Se trata del problema de ruteo de vehículos estocástico donde uno o varios componentes de la formulación son aleatorios [9], [5].

De acuerdo a lo anterior los problemas de rutas de vehículos con ventanas de tiempo, problema a resolver con el algoritmo que se programa en esta investigación pertenece al VRP homogéneo, ya que será considerando vehículos con la capacidad y características similares.

En la Tabla 2.1 se muestran los más recientes trabajos de las variantes del VRP más estudiadas según la literatura encontrada. En la primer columna de la tabla se muestra la sigla como se reconoce el problema y en la segunda columna se mencionan algunas referencias recientes.

Tabla 2.1: Relación de diferentes investigaciones del problema de ruta de vehículos.

SIGLA	**REFERENCIAS**
TSP	Miguel Cárdenas Montes, 2016 [11]
m-TSP	Rubén Iván Bolañosa, Eliana M. Toro and Mauricio Granada, 2015 [12]
CVRP	Eliana M. Toro O., Antonio H. Escobar Z., Mauricio Granada E. 2015 [13].
PTSP	Yannis Marinakisa, MagdaleneMarinakib,2009 [14]
VRPTW	Beatrice Ombuki, Brian J. Ross and Franklin Hansahr, 2006 [15] Kallehauge, Brian; Madsen, Oli B.G. ; Larsen, Jesper; Madsen, Kaj 2006 [16] Qie He, Yongjia Song, 2016 [17]
VRPSTW	Ali Kourank Beheshti, Seyed Reza Hejazi, 2014 [18]
VRPPD	Pedro Pablo Ballesteros Silva, 2016 [19]
OVRP	Ali Asghar Rahmani Hosseinabadi, Javad Vahidi, Valentina Emilia Balas, Seyed Saeid Mirkamali, 2016 [20]
OVRPTW	PP Repoussis, Tarantilis and G Ioannou Athens, 2007 [21] J. Brito, F.J. Martinez, J.A. Moreno, J.L. Verdgery 2015 [22]
Multi - trip VRP	Diego Cattaruzza, Nabil Absi, and Dominique Feillet, 2016 [23]
Multi - depot VRP	Jairo R. Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, Nilson Herazo-Padilla, 2015 [24]
SVRP	Jorge Oyola and Halvard Arntzen, David L. Woodru, 2016 [25]

2.2. Problema de ruta de vehículos con ventanas de tiempo (*VRPTW*)

Los problemas de VRP, se basan en un depósito central que cuenta con una flota de vehículos y se debe atender a un conjunto de clientes geográficamente distribuidos. El objetivo es diseñar rutas que inicien y terminen en el depósito para entregar bienes al conjunto de clientes, los cuales tiene demandas conocidas, y un costo de transporte mínimo. Cada cliente es atendido una sola vez y todos deben ser atendidos, para lo cual se asignan vehículos que llevarán la carga (demanda de los clientes que visitará) sin exceder su capacidad máxima de transporte. En la Figura 2.4 se observan los clientes representados por los puntos dispersos y los diferentes caminos que se tienen para llegar a ellos representado por las líneas, aplicando un algoritmo de solución de VRP encontramos la ruta óptima para atender todos los clientes [5].

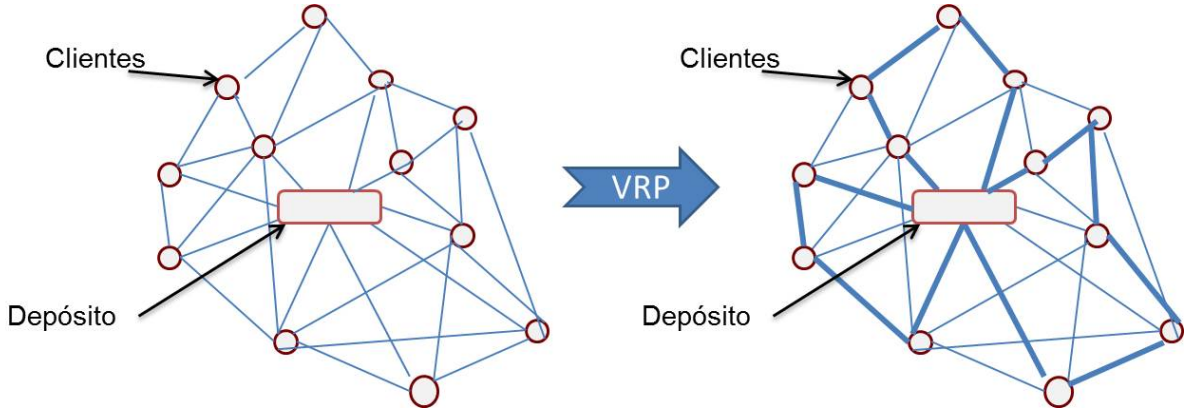


Figura 2.4: Representación de un VRP

El VRP con ventanas de tiempo considera una restricción adicional que consiste en asociar una ventana de tiempo a cada cliente, se define un intervalo en el que cada cliente debe ser atendido. Al contar con ventanas de tiempo, el costo total de ruteo considera [5] :

- **La distancia entre clientes.** Se considera una velocidad de los vehículos igual a uno, por lo que el tiempo y la distancia son iguales. Este valor se determina calculando el tiempo que hace el vehículo de transportarse de un cliente a otro.
- **El tiempo de servicio a cada cliente.** Es de suma importancia considerar el tiempo en el que se inicia el servicio a un cliente. Siempre debe ser mayor o igual al inicio de su ventana de tiempo y menor o igual al final de la misma. Si un vehículo llega a la ubicación de un cliente antes del inicio de su ventana de tiempo, debe esperar hasta el tiempo inicial para servir a ese cliente [8].
- **El tiempo de espera**. Si el vehículo llega antes del tiempo de entrada según el intervalo de tiempo asignado por el cliente, se debe de considerar ese tiempo como costo de transporte.

2.2.1. Modelo matemático

El modelo matemático que define el *VRPTW* se describe a continuación [5]: el problema *VRPTW* se define para una flota homogénea de vehículos K, un conjunto de clientes C y un grafo orientado G. El grafo contiene $\mid C+2 \mid$ vértices, de los cuáles los clientes corresponden a $C= 1,2,3,4,....n$, y el almacén está representado por dos nodos 0 y $n+1$. Al conjunto de todos los nodos se le denomina N. El conjunto de los arcos A, representa las posibles conexiones entre los nodos, todas las rutas empiezan en 0 y terminan en n+1. Cada arco $i, j, \in A$, $i \neq j$ de la red, tiene asociado un C_{ij} y una duración de viaje t_{ij} [5], [26].

El tiempo t_{ij} incluye un tiempo de servicio Ts_i al cliente i. Cada vehículo tiene una capacidad de carga Q, a cada cliente se le asocia una demanda d_i, $i \in C$ [5], [26].

Para cada uno de los clientes, el inicio de servicio debe realizarse en el intervalo de tiempo definido por el cliente llamado ventana de tiempo $[a_i, b_i]$, $i \in C$. Se especifica que si un vehículo llega demasiado temprano a la cita con el cliente, deberá esperar a que sea el inicio de la ventana temporal a_i. Se asume que todos los datos (por ejemplo Q, d_i, C_{ij}, t_{ij}, Ts_i, a_i, b_i) son números enteros conocidos y no negativos [5], [26].

Se debe asignar a cada cliente un vehículo y una secuencia de clientes para cada vehículo de tal forma que sea mínimo el costo de transporte, todo ello sujeto a las restricciones definidas anteriormente [5], [26].

EL modelo matemático tiene dos variables de decisión, X y T [26], para cada arco (i,j), donde $i \neq j$, $j \neq n+1$, $j \neq 0$, y para cada vehículo k se define X_{ij} como:

$$X_{ij} = \left\{ \begin{array}{ll} 1 & \text{si el vehículo } k \text{ viaja directamente desde } i \text{ a } j \\ 0 & \text{en cualquier otro caso} \end{array} \right\}$$

La variable de decisión T_{ij} se define para cada nodo i y para cada vehículo k e indica el momento en que empieza el servicio. En caso de que no exista el servicio, la variable no tiene significado. Se asume que $T_{0k} = 0$, $\forall$ k y que $T_{n+1,k}$ denota la llegada del vehículo k a la base.

El objetivo consiste en minimizar el costo de transporte (distancia), diseñando un conjunto de rutas, una para cada vehículo de tal forma que se cumpla con lo siguiente:

- El cliente sea atendido una sola vez
- Cada ruta empiece y termine en el depósito
- Se respeten las demandas, capacidad de los vehículos y ventanas temporales.

De tal forma que el modelo matemático se define como [5], :

Minimizar :

$$\sum_{k \in K} \sum_{i \in N} \sum_{j \in N} C_{ij} X_{ijk} \tag{2.1}$$

sujeto a:

$$\sum_{k\in K}\sum_{j\in N} X_{ijk} = 1 \qquad \forall_i \in C \tag{2.2}$$

$$\sum_{i\in C} d_i \sum_{j\in N} X_{ijk} \leq Q \qquad \forall_k \in K \tag{2.3}$$

$$\sum_{j\in N} X_{0jk} = 1 \qquad \forall_k \in K \tag{2.4}$$

$$\sum_{j\in N} X_{ihk} - \sum_{j\in N} X_{jhk} = 0 \qquad \forall_h \in C \quad \forall_k \in K \tag{2.5}$$

$$\sum_{i\in N} X_{i,n+1,k} = 1 \qquad \forall_k \in K \tag{2.6}$$

$$X_{i,jk}(T_{i,k} + t_{i,j} - V_{j,k}) \leq 0 \qquad \forall_{i,j} \in N, \forall_k \in K \tag{2.7}$$

$$a_i \leq T_{ik} \leq b_i \qquad \forall_{i,j} \in N, \forall_k \in K \tag{2.8}$$

$$X_{ijk} \in \{0, 1\} \qquad \forall_{i,j} \in N, \forall_k \in K \tag{2.9}$$

La función objetivo 2.1, indica que el costo total del recorrido deberá ser mínimo. La condición 2.2, asegura que cada cliente sea visitado solo una vez por un vehículo. Por su parte con la restricción 2.3 se asegura que cada vehículo no excede de su capacidad.

El conjunto de restricciones 2.4, 2.5 y 2.6 son ecuaciones que van a garantizar que el vehículo inicia en el nodo 0 una sola vez, abandona cualquier nodo i, $i \in C$, si y solo si se ha entrado antes a el, finalizando en el nodo $n+1$. Con la restricción 2.7 se asegura que el vehículo k no puede llegar al cliente j antes que T_{ij} cuando se viaja de i a j. La restricción 2.8 asegura que la ventana de tiempo sea cumplida, es decir que el vehículo no llegue antes de a_i, ni después de b_i y la condición 2.9 garantiza que las variables X_{ijk} son enteras.

2.2.2. Revisión de trabajos en la literatura del VRPTW

La literatura del problema del *VRPTW* inicia con casos particulares. [7] describe un sistema basado en la simulación y es desarrollado para planificar los costos aplicable a los conductores de fugones en entornos restringidos por los horarios, el objetivo principal era determinar cuando y donde ocurrían los tiempos muertos. [27] presentaron el caso del estudio de una compañía de transporte, donde emplearon una heurística manual

para aumentar la utilización de los vehículos. En este problema, las ventanas de tiempo variaban desde los 15 minutos y hasta el día entero. [28] desarrolló un algoritmo basado en la simulación Monte Carlo [29] para resolver un problema para una compañía que distribuía revistas y periódicos.

A medida que pasa el tiempo, la investigación sobre ruteo de vehículos con ventanas de tiempo avanza y la cantidad de artículos publicados aumenta.

Varios investigadores se han enfocado en el *VRPTW* usando técnicas exactas y de aproximación. El trabajo de [30] es uno de los métodos exactos más eficientes para el *VRPTW*, y ha logrado resolver pruebas de 100 clientes. Sin embargo, hasta la fecha no se ha desarrollado ningún algoritmo que logre optimizar todas las *VRPTW* con 100 clientes o más. Cabe señalar que los métodos exactos son más eficientes en las situaciones en que el espacio de la solución está restringido por ventanas estrechas de tiempo, ya que hay menos combinaciones de clientes para definir las rutas viables [31]; en 2.3.1 son descritos los métodos exactos con más detalle.

La investigación sobre la optimización combinatoria basada en metaheurísticas ha ganado popularidad especialmente desde los años noventa. Las metaheurísticas,como los algoritmos genéticos [31], [32], estrategias de evolución [33], recocido simulado [34], búsqueda tabú [35]. Estos métodos son explicados en la sección 2.3.3.

Una de las técnicas más eficaces para el *VRPTW* ha sido el desarrollo de algoritmos híbridos en dos fases que dividen la búsqueda en dos etapas: la minimización del número de rutas y los costos de viaje. Un enfoque en dos fases para el *VRPTW* se suele orientar hacia el diseño de algoritmos en dos fases, generalmente se utilizan dos procedimientos de búsqueda local distintos para explotar la minimización de la distancia de las rutas, seguida por la minimización de los costos.

En [33] introdujeron una búsqueda híbrida en dos etapas que minimiza al principio el número de vehículos usando una estrategia de evolución, y después, la distancia, es minimizada con una búsqueda tabú. En la búsqueda de dos fases, la búsqueda tabú es introducida por [36], la primera reduce a los clientes ubicados en las rutas para reducir el número total de vehículos y en la segundo realiza intercambios entre clientes reduciendo el costo de la distancia.

En [37], se introdujo una búsqueda híbrida basada en el recocido simulado y búsqueda de tabú. En [38] y [39] se comparan estudios de resultados del *VRPTW* aplicando algoritmos genéticos contra búsqueda tabú. Por su parte, [31] estudió una aplicación al *VRPTW* multi-objetivo, minimizando como primer objetivo el número de vehículos y en el segundo objetivo redujo al mínimo el tiempo total de viaje. Esto se logró mediante la adaptación del sistema de colonia de hormigas (ACS) [40].

De los artículos más citados en la literatura se han encontrado los siguientes: *Waste collection vehicle routing problem with time windows, 2005* [41]; desarrolla una algoritmo aplicado a la recolección de basura considerando ventanas de tiempo. *Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows, 2006* [15]. Presenta una solución al problema de rutas con ventanas de tiempo, como un problema multiobjetivo, en el que las dos dimensiones objetivo son el número de vehículos y el costo total (distancia), presenta una solución de algoritmo genético utilizando la técnica de clasificación de Pareto.

Durante el 2006 se publica la investigación *On the vehicle routing problem with time Windows* [16], Kallehauge concentra una revisión de literatura de las últimas tres décadas. *Time Dependent Vehicle Routing Problem with a multi ant colony system*, 2008 [42]. Porpone un algoritmo basado en la técnica de la colonia de hormigas para la solución de problemas dependientes del tiempo. *A review of dynamic vehicle routing problems*, en 2012 [43], aplica y analiza una encuesta que clasifica los problemas de enrutamiento desde la perspectiva de la calidad y evolución de la información, presenta una revisión completa de aplicaciones y métodos de solución para problemas dinámicos de enrutamiento de vehículos.

En 2014 destaca *A Novel Hibryd column generation –metaheuristic approach for the vehicle routing problema with general soft time Windows* [18]. La investigación propone una algoritmo para problemas de rutas de vehículos con ventanas de tiempo blandas, es decir que el intervalo de tiempo o ventana puede ser modificada, el algorítmo lo basa en la generación de columnas. Durante el 2016, se publicó *Vehicle Routing Problems with Time Windows and Convex Node Costs* [17], en el trabajo se desarrolla un algoritmo considerando todas las posibles soluciones de un nodo y una ruta fija para determinar la solución.

Aunque desde el 2007 se introduce la logística verde que tiene como objetivo mejorar la sostenibilidad de los procesos de producción y distribución teniendo en cuenta factores ambientales y sociales, en los últimos años se empieza a estudiar y dar importancia a la contaminación que generan los automóviles, consultar [44], [17].

En la sección 2.4 se hace referencia a los trabajos revisados en la literatura que presentan una solución del *VRPTW* con el algoritmo *GRASP*, por ser el tema de estudio del presente trabajo. El problema del *VRPTW* constituye una modelación útil de muchas situaciones reales, en la Tabla 2.2 se mencionan algunas aplicaciones de éste tipo de problemas.

Tabla 2.2: Ejemplos de aplicaciones del *VRPTW*

ÁREA ECONÓMICA	APLICACIÓN
Industria automotriz	Distribución de piezas de repuesto
Transporte de materias primas	Empresas distribuidoras
Transporte de combustible	Gas natural, gasolina,etc.
Transporte de alimento	Grandes empresas y pequeños comercios
Salud	Reparto de medicamento a farmacias.
Prensa	Distribuidora de revistas y periódicos
Banca	Reparto y recolección de dinero
Sector público	Recolección de basura, limpieza de calles, reparto de correspondencia
Agricultura	Recolección de cosechas
Industria	Suministro de mercancía y piezas entre los almacenes
Servicios	Reparación de electrodomésticos a domicilio
Educación	Ruta de autobuses escolares
Planificación	Programación de actividades
Defensa	Ruta de aviones espías, defensa militar
Transporte	Ruta de autobuses, camiones, trenes, aviones

2.3. Métodos de solución

Debido a la importancia de los problemas de optimización combinatoria en aplicaciones prácticas, científicas e industriales se han desarrollado múltiples métodos para resolverlos, tal como se muestra en la Figura 2.5.

Actualmente existen diferentes métodos para solucionar el VRP y las variantes existentes, agrupándose en exactos y aproximados. Varias propuestas de algoritmos exactos son apropiadas en problemas pequeños, pero dado el alto costo computacional no son adecuadas para problemas mayores y en estos casos se usan algoritmos aproximados.

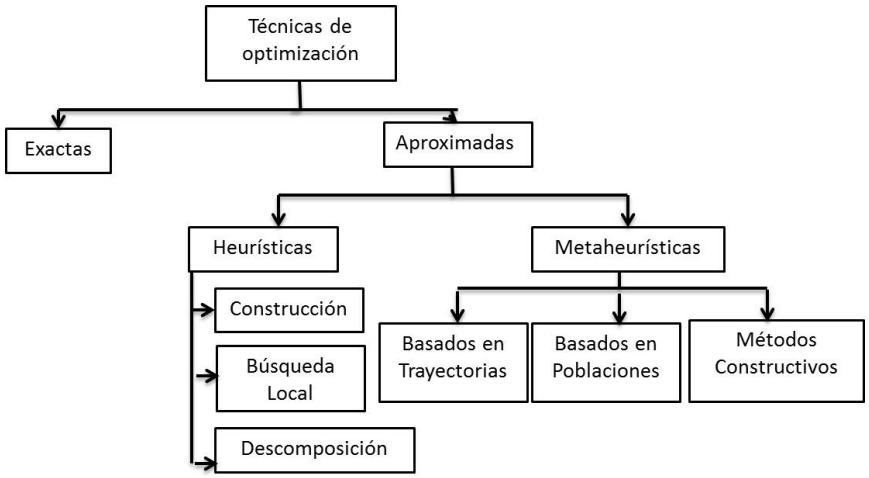


Figura 2.5: Clasificación de técnicas de optimización

2.3.1. Algoritmos exactos

Este tipo de algoritmos es utilizado para resolver problemas de instancias con una reducida cantidad de clientes, tratan de asegurar una solución óptima con el riesgo de emplear un tiempo computacional alto, a veces no disponible.

Entre los métodos exactos que más destacan podemos mencionar: los ***métodos de búsqueda directa de árbol***: con este método la búsqueda se realiza sobre todos los nodos de un árbol de acuerdo a criterios específicos propios de cada método, destacando tres algoritmos, (1) el algoritmo de asignación de cota inferior, que consiste en asignar una cota inferior que permite disminuir el número de vehículos requeridos para visitar todos los vértices. [45], [9].

Por su parte, (2) el algoritmo de ramificación y acotamiento consiste en recorrer cada nodo del árbol desde el nivel superior hacia la base del árbol y los nodos terminales. Resolviendo en cada nodo un programa lineal y determinando que nodos pueden eliminarse. Un nodo se elimina (junto con sus descendientes) si no existe una solución factible; pero si existe solución factible se convierte en una cota inferior. El algoritmo termina cuando todos los nodos han sido revisados y la solución óptima es la de mayor cota inferior[26].

(3) El árbol del centro de k-grados (k-degree center tree algorithm) trabaja con un número fijo de vehículos, una solución factible en el conjunto de aristas se divide en cuatro subconjuntos que son: las aristas que no pertenecen a la solución, las aristas que forman el árbol, las aristas que inciden en el primer vértice y las aristas que no inciden en el primer vértice. Estos subconjuntos se traducen en restricciones en el modelo y

la solución objetivo consiste en sumar el costo de todas las aristas en la solución [45] [9].

Programación dinámica: En este método se considera un número fijo de m vehículos, se encuentra primero el costo mínimo alcanzable utilizando k vehículos, teniendo en cuenta la función del costo en la longitud de una ruta de vehículos a través de todos los vértices del subconjunto, luego encuentra el costo de todos los subconjuntos de vértices con m vehículos [45].

Programación lineal y entera. El método de partición y generación de columnas se considera un conjunto factible de rutas y un coeficiente binario que es igual a uno si y solo si, determinado depósito pertenece a una ruta. También se tiene en cuenta el costo óptimo de una ruta y una variable binaria que es igual a uno si y solo si, esa ruta es utilizada en la solución óptima. El algoritmo desarrollado está basado en una formulación que garantiza la solución óptima en un número finito de pasos, si se ejecuta hasta finalizarlo. La formulación no exige que los vehículos sean idénticos [45], [9].

Algoritmos aproximados

Autores contemporáneos los clasifican a su vez en heurísticas y metaheurísticas. En ocasiones se consideran heurísticas con procedimientos de resolución aproximadas, otros investigadores engloban métodos heurísticos y métodos metaheurísticos bajo el concepto de optimización heurística, sin embargo se pueden mencionar características que nos permiten diferenciarlos [26].

Heurísticas son procedimientos simples, no aseguran encontrar la solución óptima pero si una muy cercana a través de una exploración limitada del espacio de solución. Las soluciones obtenidas son aceptables y tienen una complejidad algorítmica baja, pueden ser mejoradas a través de otros procedimientos sofisticados pero a mayor costo computacional [46].

Por su parte las metaheurísticas emulan los procedimientos de la naturaleza o la inteligencia artificial como la evolución biológica, comportamiento de algunos insectos, mecánica, estadísticas, entre otras [26].

2.3.2. Métodos heurísticos.

Los métodos heurísticos permiten encontrar soluciones que tienden a ser muy cercanas al óptimo, sin embargo es difícil determinar que tan cercana esta la solución del óptimo. Por lo anterior se suele utilizar comparaciones con métodos exactos u otros patrones a fin de medir el desempeño del algoritmo. Su ventaja más notable con respecto a los métodos exactos es que permite darle escalabilidad a los problemas que se estudian y son aplicables a problemas difíciles ya que permiten encontrar soluciones muy cercanas al óptimo para instancias muy grandes sin que su tiempo de ejecución se incremente

demasiado. Otra característica inherente a las heurísticas es que son aplicables a un problema en particular, ya que se construyen a partir de la estructura misma del problema. [47].

Una de las razones más importantes para su uso es que en ocasiones no existen métodos exactos que proporcionen una solución, y es factible entonces contar con una solución aproximada, cuando las circunstancias del costo en tiempo, almacenamiento y memoria son importante. En ocasiones se utiliza como base para aplicar un método exacto [26].

Algunas de las técnicas heurísticas que se mencionan son los métodos constructivos, métodos descomposición, búsqueda local, de reducción y de manipulación de modelo [26].

1. **Métodos constructivos:** Dentro de estos métodos se pueden mencionar los algoritmos de ahorros, los cuáles consisten en combinar rutas e irlas modificando de acuerdo a la que permita algún ahorro en distancia [9], [46].

   Otro de los métodos constructivos son las heurísticas de inserción, las cuales crean soluciones insertando sucesivamente clientes en las rutas, en cada iteración se tiene una solución parcial, cuyas rutas solo visitan un subconjunto de los clientes y después se selecciona un cliente no visitado para insertarlo en la última ruta creada [9]. Ambos métodos constructivos mencionados anteriormente son ampliamente conocidos dentro del campo de la investigación de operaciones [9].

2. **Métodos descomposición:** Se basa en el principio divide y vencerás, divide el problema general en otros más pequeños para obtener una solución general [26].

3. **Búsqueda local.** Con el fin de mejorar una solución se aplica un algoritmo de búsqueda local. El procedimiento se basa en definir un conjunto de soluciones vecinas y partir de una solución primaria para luego remplazarla por una solución vecina con menor costo. El procedimiento se repite hasta que no pueda mejorar la solución. Se exploran los vecindarios evaluando la contribución a la función objetivo de cada solución tendiendo a maximizar o minimizarla, según corresponda [9], [46], [26].

La Figura 2.6 muestra el procedimiento de los algoritmos de búsqueda local, se representa como las soluciones se desplazan por el espacio de búsqueda a través de vecindarios construidos con las mejores soluciones encontradas, donde los círculos representan los vecindarios y los puntos rojos los clientes [47].

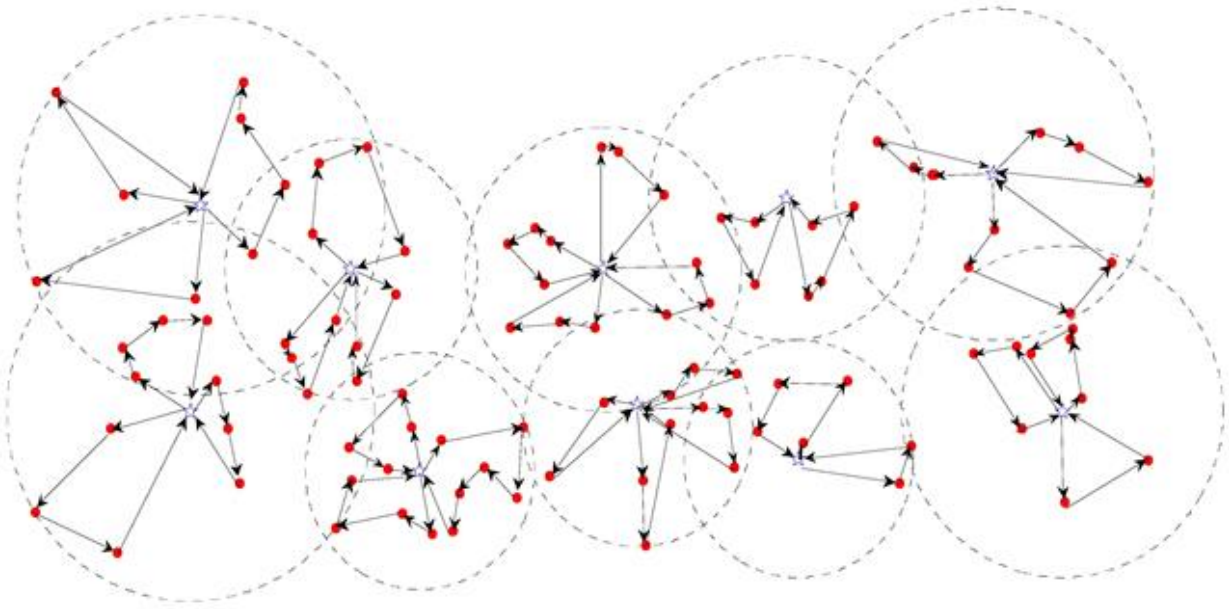


Figura 2.6: Representación de búsqueda local en vecindarios, [48]

2.3.3. Métodos metaheurísticos

Las metaheurísticas se introducen en la década de los 80 y se caracterizan por que realizan un procedimiento de explorar el espacio de soluciones para encontrar soluciones de mejor calidad, aplicando procedimientos que modifican las soluciones intermedias de acuerdo a la función objetivo [9].

Las metaheurísticas más explorada son el recocido simulado, búsqueda tabú, algoritmos genéticos, algoritmos de hormigas, búsqueda de vecindades y *GRASP* [9], [26]. Existen marcadas diferencias entre las metaheurísticas conocidas, sin embargo todas se basan en intensificar y diversificar la búsqueda con el fin de mejorar la solución con respecto a la función objetivo evitando los óptimos locales.

- ***Recocido simulado (Simulated Annealing).*** Es un procedimiento introducido por Kirkpatrick, Gelatt y Vecci [49] y ha sido empleado con frecuencia para resolver problemas combinatorios. La idea surge del proceso físico conocido como recocido, en el cual se eleva la temperatura de un sólido hasta el punto que se vuelve líquido, a continuación la temperatura se disminuye de forma paulatina para obtener una estructura cristalina sin defectos (estado basal). Es un método de optimización combinatoria basado en técnicas de aleatorización, se basa en la analogía entre el recocido de sólidos, tal como se describe en la teoría de la física y la optimización de grandes problemas combinatorios. Con este método se pueden

obtener soluciones cercanas a un óptimo. El método del recocido se utiliza en la industria para obtener materiales más resistentes, o más cristalinos, en general, para mejorar las cualidades de un material [50].

El proceso consiste en derretir el material. En esa situación, los átomos adquieren una distribución aleatoria dentro de la estructura del material y la energía del sistema es máxima. Después se desciende la temperatura muy lentamente por etapas, dejando que en cada una de esas etapas los átomos alcancen una configuración óptima para esa temperatura, al final del proceso, los átomos forman una estructura cristalina altamente regular, el material alcanza así una máxima resistencia y la energía del sistema es mínima [50].

- ***Búsqueda tabú.*** Alrededor del año 1986, se iniciaron varios trabajos sobre éste método de solución, oficialmente fueron introducidos en 1989 por Fred Glover [51], es una técnica basada en la inteligencia artificial para resolver problemas combinatorios de gran dificultad, una de las principales características es utilizar una memoria flexible de búsqueda, la cual se utilizará para modificar, restringir y expandir sobre el entorno de vecindad [26].

  Además búsqueda tabú permite elegir una solución vecina que no es mejor que la actual con el fin de evitar óptimos locales y la característica primordial del método es que guarda una lista tabú de movimientos o soluciones para evitar que se cicle [26].

- ***Algoritmo de optmización por colonia de hormigas (Ant algorithms).*** La metaheurística ACO fue introducida por Marco Dorigo [52] y a partir de dicho desarrollo se han generado diversas propuestas que buscan mejorar el funcionamiento del algoritmo original. La técnica se basa en el comportamiento de los insectos, específicamente están inspirados en la estrategia que usan las colonias de hormigas en la búsqueda de alimentos. Cuando una hormiga encuentra alimento va depositando una sustancia llamado feromona por el camino que depende de la calidad del alimento y la longitud [26]. Las hormigas siguen el camino con el rastro de la feromona con mayor cantidad, lo que a su vez provoca mejores caminos, que serían los de menos tiempo y por donde transiten la mayor cantidad de hormigas [26].

- ***Búsqueda de vecindades (VNS, variable neighborhood search).*** Este método fue presentado por Mladenovic y Hansen en 1997 [53], el método principalmente consiste en evaluar el entorno y encontrar una mejora, entonces se introduce un segundo operador con la esperanza de salir de los óptimos locales [26].

- ***Algoritmos genéticos.*** Fueron introducidos en 1970 por John Holland [54] inspirados en el proceso evolutivo de los seres vivos para resolver problemas de

optimización y búsqueda [9], [26].

La evolución se basa en los cromosomas que es donde se codifica la información de los seres vivos, ésta información cambia en cada generación. Las características de esta técnica son: la evolución que opera en los cromosomas, la selección de los cromosomas se va realizando con aquellos que presenten mejores estructuras en el proceso de reproducción, se inicia con los progenitores pero se deben considerar las mutaciones que pueden alterar estos códigos, y no se almacena memoria por lo que solo se va considerando la generación anterior [9], [26].

Basado en lo anterior los algoritmos genéticos simulan este proceso con el fin de obtener soluciones que se van cruzando, y mejorando así la solución hasta encontrar un criterio de parada.

- **GRASP** ***(greedy randomized adaptative search procedures).*** Este método fue desarrollado a finales de la década de 1980 por Feo y Resende, 1989 [55], es una técnica desarrollada para estudiar problemas de alta complejidad combinatoria, cuya traducción literal sería "procedimientos de búsquedas ávidos, aleatorios y adaptativos"[26].

  El método se desarrolla en una fase de construcción y una de mejora donde se aplica una búsqueda local y se caracteriza en construir soluciones de calidad que son utilizadas posteriormente para obtener otras soluciones mejores. Durante la fase de construcción se ordenan los datos de forma voraz y adaptativa, en acorde con la función objetivo, seleccionando en la lista de candidatos aleatoriamente al ir construyendo la solución [26]. La segunda fase tiene como objetivo mejorar la solución obtenida en la fase de construcción. El procedimiento se puede describir con el siguiente pseudocódigo:

Algoritmo 1: Forma general del algoritmo GRASP

D= instancia de prueba, α, condición de parada;
mientras *(no se cumpla la condición de parada)* **hacer**
    S= Algoritmo Constructivo (D);
    S= Búsqueda local (S);
    Registrar mejor solución (S, MS);
Regresar la mejor solución (MS);
Fin $GRASP$;
D= datos de entrada, S = solución, MS= mejor solución.

2.4. *GRASP* aplicado al *VRPTW*

Al realizar una revisión de diferentes bases de datos como Elsevier, Springer, IEEE entre otras se encontró que actualmente se ha aplicado la metodología del Greedy Randomized Adaptative Search Procedures (*GRASP*) para solucionar problemas de rutas de vehículos con ventanas de tiempo, *VRPTW*, en 14 investigaciones.

Analizando y comparando las investigaciones encontradas en la literatura, específicamente las que utilizaron el algoritmo *GRASP* en la fase de construcción de soluciones, se ha concluido que el método de construcción utilizado en el presente trabajo para la solución de problemas del *VRPTW* difiere en los métodos de ordenamiento y en la aplicación de diferentes métodos de mejora, tal como se explica en el siguiente capítulo.

En la Tabla 2.3, se hace referencia al método de Solomon [56], quien diseñó un algoritmo que se basa en construir las soluciones en forma secuencial, es decir una ruta a la vez. Su método inicia con dos listas que agrupan por un lado los vértices mas alejados del depósito y la otra lista agrupa al resto; inicia con un cliente de la lista de los clientes mas alejados, quien encabeza la primer ruta y después hace un recorrido por la otra lista insertando en la ruta los clientes que cumplan con la restricción de las ventanas de tiempo y minimicen la distancia, y que requieran el servicio más temprano, una vez que ya no sea posible insertar clientes, se inicia nuevamente un cliente de los mas alejados del depósito, y así sucesivamente hasta que las listas estén con 0 vértices.

Lim y Wang [57] construyen la solución con el método *GRASP* creando múltiples soluciones heurísticas, en cada iteración inicia de forma diferente, aleatoriamente con alguna de las cuatro formas que se describen a continuación:

- Seleccionando primero un vértice, e intenta ir insertando a los clientes que cumplan con las restricciones de tiempo. La inserción de clientes continua hasta que ya no sea posible insertar ningún cliente en esa ruta.
- El cliente mas cercano al depósito primero, construye una ruta para cada vehículo uno por uno, insertando el cliente que cumpla con la restricción de tiempo y distancia menor, al último cliente de la ruta, repitiendo la inserción hasta que no sea posible insertar ningún cliente a la ruta en construcción.
- Otra forma de construir fue utilizando otro procedimiento de ordenación, basado en los clientes que presentan menor tiempo de espera para dar el servicio. Construyendo ruta por ruta.
- La siguiente ruta la inicia con aquél cliente que este más lejos, de allí continua insertando los vértices necesarios hasta terminar de construir la ruta.

Zhiye Li [58] basa la construcción del *GRASP* introduciendo seis formas de inicializar soluciones, las primeras cuatro son iguales que Lim y Wang [57], la quinta y sexta utiliza métodos aleatorios, seleccionando clientes al azar e irlos insertando en rutas donde

proporcione la mínima distancia en el método cinco y finalmente en el seis construye rutas de forma aleatoria y al momento en que ya no es posible insertar clientes bloquea las rutas. En cada iteración se construye de una de las seis formas mencionadas anteriormente y además modifica el método de *GRASP* ya que reutiliza la solución de la iteración anterior como la entrada para la construcción ganadora aleatoria en la iteración actual.

Yong Mao [59], utiliza del *GRASP* la parte voraz con la finalidad de construir soluciones a las cuales después les aplica un método genético y finalmente el método PSO (Optimización de enjambre de partículas).

En la Tabla 2.3 se detallan las investigaciones encontradas en la literatura que resuelven el problema del *VRPTW* aplicando el algoritmo *GRASP*, donde:

- Nombre del Artículo: Indica la denominación del artículo
- Método Utilizado: Es el método que se utilizó para obtener la solución
- Diferencia: Se señala la diferencia de la metodología utilizada en el artículo con el algoritmo que se aplica en la presente investigación.

Una vez realizada la investigación exploratoria de los problemas que resuelven el *VRPTW* con el algoritmo *GRASP*, las contribuciones que se presentan en este trabajo son las siguientes:

- Se programa un algoritmo no descrito en la literatura que combina diferentes formas de ordenamiento en la fase de construcción de soluciones y se aplican tres métodos de mejora para la solución del *VRPTW* basado en la metodología *GRASP*.

- Se realiza una calibración de los parámetros de inicio para la construcción de soluciones, destacando que los trabajos descritos anteriormente utilizan una lista de candidatos de 3 clientes [60], [61] y en el algoritmo desarrollado en el presente trabajo, las soluciones de mejor calidad fueron proporcionadas con una lista de candidatos variable, dependiendo del valor de α establecido en 0.90, de acuerdo a la Fórmula 3.2.
  Declarando así que de acuerdo al teorema de optimización *No Free Lunch* (NFL, por su sigla en inglés) [62], que establece que no existe un vector de parámetros óptimo que de solución a todos los problemas de optimización, por lo que no se debe generalizar. La selección del vector de parámetros en los algoritmos metaheurísticos depende del problema y es una cuestión importante para asegurar la calidad de soluciones, por lo que se requiere una estrategia sistemática y robusta del diseño de parámetros para realizar con precisión y eficiencia la calibración [63].

Tabla 2.3: Investigaciones que aplican *GRASP* para solucionar problemas de *VRPTW*.

Nombre del Artículo	**Método**	**Diferencia**
A *GRASP* for *VRPTW*, [60]	*GRASP*	La fase de construcción la realiza determinando el mínimo de rutas necesarias y después va insertando clientes de acuerdo al método de Solomon y aplica el 2 opts.
Grasp with a new local search scheme for *VRPTW*, [61]	GRASP	Construye soluciones determinando rutas e insertando clientes de acuerdo a la distancia máxima al depósito y al cliente que requiera el servicio más temprano, en la búsqueda local elimina rutas sobrantes y aplica la heurística 2 opts.
A Smoothed Dynamic Tabu Search Embedded *GRASP* for m-VRPTW, [57]	*GRASP* TABU	Construye soluciones de dos formas: Crea diferentes parámetros de inicialización, y una vez que tiene la mejor solución vuelve a considerarla como base para construir nuevamente otra ruta. En la búsqueda local se utilizan varias consideraciones.
Improved *GRASP* with Tabu Search for VRP with Both TW and Limited Number of Vehicles, [58]	*GRASP*	La fase de construcción la realiza con diferentes formas de inicializar las ruta y en búsqueda local elimina una ruta y acomoda los clientes en otra.
A Reactive Greedy Randomized Variable Neighborhood Tabu Search for the *VRPTW*,[64]	*GRASP* TABU	El mecanismo de construcción *GRASP* propuesto utiliza un esquema de construcción de solución de inserción paralelo junto con una función codiciosa basada en penalidades que combina de manera ponderada un conjunto de criterios, la búsqueda local aplica el VNS y búsqueda Tabú
A *GRASP*–VNS Hybrid for the Fuzzy *VRPTW*, [65]	*GRASP* VNS	No especifica como construyó la solución en la primera fase del *GRASP*, en la búsqueda local utiliza VNS.
Solving *VRPTW* with Hybrid Evolutionary Algorithm, [59]	Algorítmo Genético y *GRASP*	Construye la ruta con el método *GRASP* de una manera voraz, se escogen las mejores soluciones para aplicar el PSO, seleccionando dos padres y aplicar algoritmo genético.
A *GRASP* × ILS for the *VRPTW*, synchronization and precedence constraints, [66]	*GRASP*	Aplica el *GRASP* iniciando una ruta para cada vehículo, construyendo rutas de manera paralela, iniciando con los clientes que por tiempo se deben de visitar primero, deteniéndose cuando todos los clientes ya han sido asignados, en la búsqueda local utiliza las siguientes estrategias: evalúa si es posible reubicar los clientes en una misma ruta, se evalúa acomodar un cliente de una ruta en otras rutas, intercambia 2 clientes en una misma ruta, siempre buscando minimizar la distancia.
An ACO hybrid metahuristic for close - open *VRPTW* and fuzzy constrints, [22]	ACO *GRASP* VNS	Construye la solución de forma aleatoria y siguiendo la técnica ACO y en búsqueda local aplica la técnica del VNS.
A hybrid *GRASP-VNS* for ship routing and scheduling problem with discretized time windows, [67]	*GRASP* VNS	Se construye una solución realizando una ruta a la vez, con el objetivo de disminuir la flota de buques y la distancia, en la búsqueda local se aplica el VNS, en 3 fases , realiza el movimiento, evalúa la F.O. y después se hace el cambio.
VRPTW and multiple service workers: a systematic comparison between ACO and *GRASP*, [68]	ACO *GRASP*	El procedimiento de *GRASP* que utiliza es la técnica de Solomon y la búsqueda local determinista que consiste en reubicar y cambiar los clientes.

Capítulo 3

Desarrollo de Investigación

A continuación se describe la metodología en la que se sustenta el presente proyecto, que es la propuesta por Yaghini, 2012 [69] denominada DIMMA (Metodología de Diseño e Implementación para Algoritmos Metaheurísticos). Después se describe en que consiste el algoritmo aplicado a la solución del $VRPTW$ y las búsquedas locales que se modelaron y programaron para obtener los resultados mostrados en el siguiente capítulo.

3.1. Metodología DIMMA

La metodología DIMMA se clasifica en tres fases secuenciales, las cuales consideran varias etapas tal como se muestra en la Figura 3.2. En la parte de identificación, se llevan a cabo las siguientes etapas:

- Declaración del problema. Se inicia con la definición y delimitación del problema; se sugiere un modelo matemático, siempre que sea posible; se especifica como estará estructurado el problema, de que forma se realizará la selección de estrategias de solución y se definen las medidas de desempeño.

- Definición de objetivos. Es importante definir de forma clara los objetivos que se establecen en la implementación de un algorítmo metaheurístico, algunos ejemplos son: (1) reducir el tiempo de búsqueda de solución en comparación con métodos exactos u otra metaheurística, (2) mejorar la calidad de las soluciones, (3) robustez en las instancias, (4) resolver problemas a gran escala, (5) facilidad de implementación, (6) facilidad para combinar con otros algorítmos y mejorar el rendimiento, (7) flexibilidad para resolver otros problemas de optimización, (8) en la selección de instancias, el método de solución y la definición de medidas de rendimiento deben realizarse de acuerdo con las metas seleccionadas.

- Selección de instancias de prueba. Las instancias que se seleccionen deben ser representativas al problema de estudio, deben ser diversas en términos de tamaño, de su complejidad y su estructura, ya que esto será de gran utilidad en el análisis del rendimiento del algoritmo metaheurístico construido.

Las instancias que se pueden utilizar son instancias reales y construidas. Las primeras son ejemplos prácticos de aplicaciones del mundo real, es decir, solucionan un ejemplo de la vida real; las segundas son las que se encuentran en la literatura utilizadas con más frecuencia en las experimentaciones.
Una vez que se seleccionan las instancias, deben dividirse en dos subconjuntos; uno para ajustar los parámetros del algoritmo y el segundo para evaluar el rendimiento del algoritmo con los parámetros ajustados. Las instancias de ajuste deben ser representativas de toda la clase de instancias que el algoritmo resolverá. Por lo tanto, dado que las instancias de ajuste son representativas de otras, el rendimiento del algoritmo en estos dos subconjuntos de instancias debe ser similar entre sí.

En el planteamiento se consideran los siguientes puntos:

- Selección de estrategias de solución. En esta etapa se justifica la metaheurística, para ello se revisan los métodos de solución existentes que resuelven el problema a solucionar, Una vez seleccionado el método a utilizar para la solución del problema se deberán especificar los parámetros del algoritmo que resuelven el problema. Esta especificación puede ser útil en el paso de seleccionar la estructura de datos y diseñar el algoritmo.

- Definición de indicadores de desempeño. En este punto, se define como se medirá el rendimiento del algoritmo. Se debe seleccionar las medidas apropiadas de acuerdo con los objetivos seleccionados para las metaheurísticas que tratan de encontrar una solución cercana a la óptima en un tiempo razonable, tanto la calidad de la solución como el tiempo computacional son los dos indicadores principales del desempeño.

  La calidad de la solución y el tiempo computacional son los dos indicadores principales que indican el desempeño en las metaheurísticas, los indicadores para evaluar la efectividad incluyen la calidad de las soluciones, el esfuerzo computacional y la robustez.

  La calidad de las soluciones se basa en medir la distancia (Gap) a una de las siguientes soluciones: solución óptima, solución inferior o superior, solución mejor conocida y solución real implementada. Figura 3.1 .

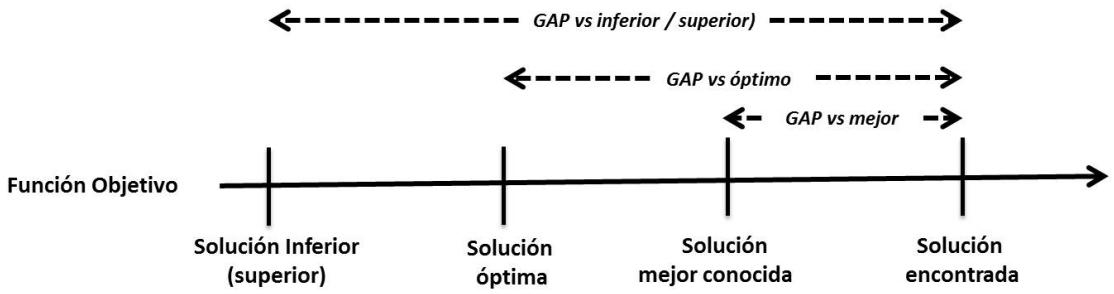


Figura 3.1: Evaluación del desempeño de la calidad de las soluciones en un problema de minimización

- Selección de estructura de datos. Antes de empezar a trabajar en el diseño del algoritmo, es necesario seleccionar una estructura de datos adecuada. La estructura de datos es un esquema para organizar la información en la memoria, para una mejor eficiencia del algoritmo, como por ejemplo archivos, listas, árboles y tablas. Es importante seleccionar adecuadamente la estructura de datos ya que eso nos facilitará la programación, si se selecciona una estructura de datos errónea para un algoritmo, la implementación del mismo podría ser demasiado lenta.

- Diseño del algoritmo. Se refiere a la forma como se especifica el programa, puede ser en su forma más sencilla (lenguaje natural), o como un pseudocódigo que es una combinación de lenguaje natural y de programación. Otra forma es utilizando un diagrama de flujo.

  El algoritmo se analiza de acuerdo a cuatro factores, la complejidad (el número de pasos de pseudocódigo necesarios para especificar el algoritmo), la eficiencia espacial (el uso de espacio o memoria de un algoritmo), la simplicidad (a medida de que el algoritmo sea simple, será más fácil programarlo) y por último la generalidad (es la aplicabilidad de un algoritmo a una amplia gama de insumos).

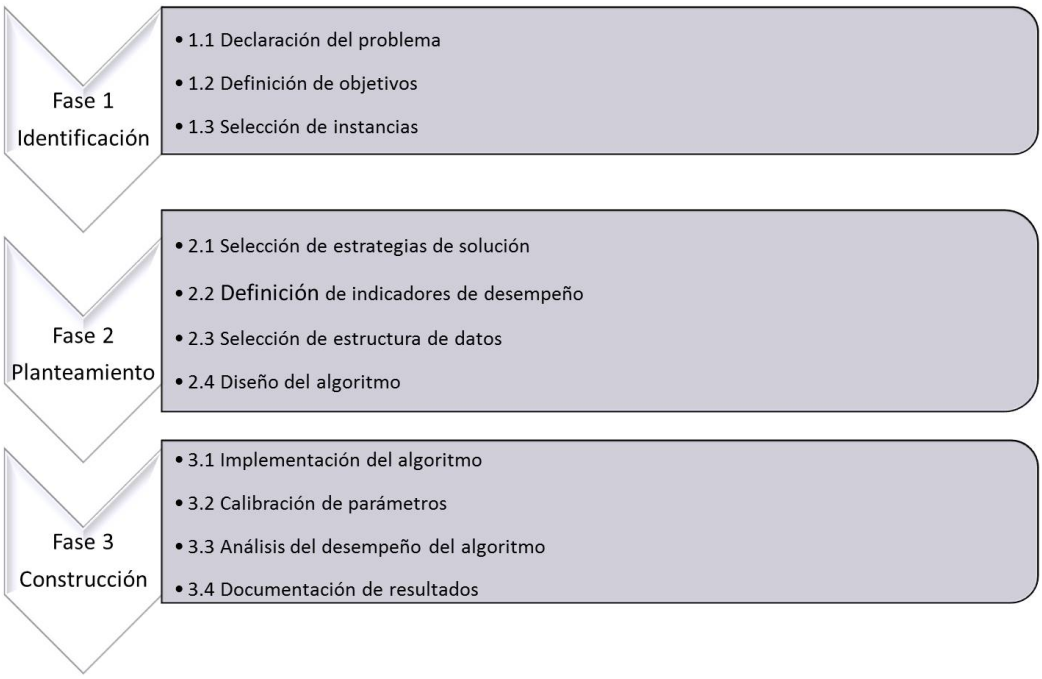


Figura 3.2: Pasos en cada fase de la metodología DIMMA

Finalmente, en la construcción, se llevan a cabo las siguientes etapas:

- Implementación del algoritmo. El algoritmo se implementa mediante un lenguaje de programación adecuado.

- Calibración de parámetros. Los parámetros son medidas descriptivas de toda una población que se utilizan como las entradas para un algoritmo metaheurístico. Las metaheurísticas son sensibles al valor de sus parámetros, es por esto que es de importante ajustar los parámetros.

  En la mayoría de los trabajos de investigación en el campo de la metaheurística, los parámetros se ajuntan a mano, es decir utilizando un procedimiento de ensayo y error. Este enfoque tiene varias desventajas, tales como: tiempo, trabajo intensivo, y necesita un profesional con habilidades especiales.

  Los valores apropiados para los parámetros dependen de tres factores principales: el problema en cuestión, la instancia del problema y el tiempo de búsqueda requerido.

  Existen dos estrategias diferentes para el ajuste de parámetros:

  1. Ajuste offline. Se hace referencia aquellos parámetros que se establecen antes de la ejecución de las metaheurísticas y no se actualizan durante la ejecución. En esta estrategia, establecer los parámetros no es garantía que sean los valores óptimos. Para superar este problema, se puede utilizar el diseño

de experimentos (DOE).
Al realizar un DOE, se determinan los factores (parámetros) con diferentes niveles (valores potenciales de los parámetros). Con un diseño factorial completo, se puede obtener el mejor nivel de cada factor, pero este trabajo requiere un alto tiempo computacional. Sin embargo, existen varios enfoques de DOE que reducen el número de experimentos. Al aplicar un DOE se refiere a la planificación del experimento, se determinan los datos apropiados a utilizar, resultando en conclusiones válidas y objetivas. Una de las desventajas de la la estrategia de ajuste offline es que el valor o los parámetros son fijos y no pueden cambiar durante la búsqueda.

2. Ajuste online. Actualiza los parámetros de forma dinámica (una actualización aleatoria o determinista de los valores de los parámetros sin tener en cuenta el proceso de búsqueda) o adaptativamente (cambia los valores según el progreso de la búsqueda utilizando la memoria de la búsqueda). En los enfoques adaptativos, los parámetros se codifican en la representación de soluciones y como resultado, al ir modificando la solución, el valor de los parámetros se cambiará.

- Análisis del desempeño del algoritmo. Durante esta etapa, se recolectan los resultados experimentales y se analiza el desempeño de algoritmos metaheurísticos con pruebas estadísticas, tales como test t, y modelos ANOVA en caso de una comparación de más de dos algoritmos.

  Estas pruebas estadísticas se utilizan para determinar si la conclusión obtenida se debe o no a un error de muestreo. La selección de una herramienta de prueba de hipótesis estadística determinada se realiza de acuerdo con las características de los datos. En general, no basta con analizar un algoritmo basado únicamente en el enfoque teórico, de modo que el análisis empírico del desempeño es una tarea necesaria que debe realizarse de manera justa.

- Documentación de resultados. En esta etapa final de la metodología DIMMA, los resultados deben ser analizados e interpretados basados en los objetivos planteados y los indicadores de desempeño definidos. Generalmente, las herramientas gráficas son más comprensibles comparado con la presentación de resultados utilizando tablas.

3.2. Algoritmo *GRASP* aplicado al *VRPTW*

Con la finalidad de aplicar una metaheurística para solucionar el *VRPTW* se realizó una exploración en las metaheurísticas estudiadas en la literatura presentada en el capítulo anterior que resuelven este tipo de variante del VRP, y se optó por aplicar el *GRASP*.

La metaheurística *GRASP* fue seleccionada como una estrategia para obtener soluciones iniciales diversas y de buena calidad, por su capacidad de exploración, además en su

etapa de construcción genera diversas alternativas y este hecho es una ventaja para las aplicaciones de problemas de rutas de vehículos donde pueden darse eventos inesperados, y evitar estancarse en óptimos locales. Otro factor importante que se consideró es que comparado con otras metaheurísticas *GRASP* parece ser competitivo con respecto al número de parámetros para ajustar, (Sólo dos: el tamaño de la lista de candidatos y el criterio de parada), la calidad de las soluciones y la baja complejidad de implementación [70].

Una vez que se obtengan soluciones, se realizará una comparación con las instancias de Solomon 1987 [56] y así probar la efectividad del algoritmo que se modela en este trabajo.

3.2.1. Instancias

Las instancias son un conjunto de datos que se usan para medir los resultados obtenidos al aplicar los métodos de optimización; se usan con frecuencia en el campo científico para comparar resultados propios con los de algunos otros de la literatura. Están asociadas a características de los problemas y se usan en el planteamiento específico de un modelo. El conjunto propuesto por Solomon [56] es un valioso recurso y punto de referencia en la validación de técnicas heurísticas del *VRPTW*. Como se detalló en la Tabla 2.3, las instancias de Salomón [56] son las más utilizadas para comparar la eficiencia de los algoritmos aplicados a *VRPTW*. En [71], se muestran los resultados mejor conocidos.

El conjunto de pruebas consta de 168 instancias, las cuales se presenta con 25, 50 y 100 clientes, con las siguientes capacidades de carga de vehículos y notación:

- Las de capacidad de 200 son las de notación C1(19), R1(12) y RC1(8).
- Las que reportan capacidad de 700 son las C2(8).
- Finalmente las de capacidad de 1000 son las que presentan la nomenclatura R2(11) y RC2 (8).

El número entre paréntesis muestra la cantidad de instancias que existen en ese grupo. Los grupos de instancias difieren entre sí por el ancho de las ventanas de tiempo, medido entre la diferencia del inicio y fin de la ventana; como se muestra en la Tabla 3.1. Algunos clientes tienen ventanas de tiempo estrecho, identificada con 1, mientras que otros clientes tienen ventanas de tiempo amplias, identificadas con el número 2. Existen diferencias de igual forma en la ubicación de los clientes según sus coordenadas y densidad de ventanas que refiere al porcentaje de clientes que no inician en el depósito.

En el banco de instancias de tipo R1 y R2 las coordenadas geográficas son generadas aleatoriamente (Random). Los grupos C1 y C2 (clúster), comprenden los problemas

en que los clientes están agrupados por zonas o por clústers. Los grupos RC1 y RC2 (Random-Clúster) son problemas de semiclúster, es decir, que son una mezcla de los dos anteriores.

El conjunto de los problemas 1: R1, C1, y RC1 son de corto horizonte de programación. Los problemas 2: R2, C2, RC2 son de largo horizonte de programación, es decir que muchos clientes pueden ser servidos por un mismo vehículo (hasta 30 clientes) generando menos rutas que el conjunto de problemas 1 (entre 5 y 10 clientes).

En una instancia pueden existir clientes cuyo inicio de ventana tiene valor cero. El conjunto de clientes que tienen inicio de ventana de tiempo diferente de cero, son clientes que poseen corto horizonte de tiempo para programarlos en una ruta parcial. Para una instancia de 100 clientes con densidad de ventana del 25 %, existe un conjunto de 25 clientes con inicio de ventana diferente de cero. Para las instancias de largo horizonte existe densidad de ventana bajas de 25 y 50 %. Para la mayoría de las instancias de corto horizonte de programación, las densidades de ventana, son del orden del 75 % y 100 %. Consultar Tabla 3.1 donde se identifica geográficamente a los clientes como:

- C = Clouster, es decir distribuidos por grupos.
- R= Random, distribuidos aleatoriamente.
- RC = Distribuidos mixtos, es decir en grupo y aleatorios

Tabla 3.1: Tabla de clasificación de instancias

Clasificación	Característica		
Distribución geográfica	C	R	RC
Densidad de ventana	25 %	50 %	75 %
Ancho de ventana	1	2	
Horizonte de programación	1	2	

En la Figura 3.3 se puede observar un ejemplo de una instancia de Solomon. La instancia mostrada es la C101.txt, y proporciona los siguientes datos:

- Vehicle number, número de vehículos = 25
- Capacity, capacidad =200
- Customer Cust. No.= columna que representa el número de clientes.
- XCOORD = Coordenada X, donde ubica a los clientes.
- YCOORD = Coordenada en Y
- DEMAND = demanda

- READY TIME = Inicio de la ventana de tiempo
- DUE DATE = Final de la ventana de tiempo
- SERVICE TIME = Tiempo de servicio que ocupa cada cliente.

C101.txt

C101

VEHICLE
NUMBER CAPACITY
25 200

CUSTOMER

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	652	721	90
13	22	75	30	30	92	90
14	22	85	10	567	620	90
15	20	80	40	384	429	90
16	20	85	40	475	528	90
17	18	75	20	99	148	90
18	15	75	20	179	254	90
19	15	80	10	278	345	90
20	30	50	10	10	73	90
21	30	52	20	914	965	90
22	28	52	20	812	883	90
23	28	55	10	732	777	90
24	25	50	10	65	144	90
25	25	52	40	169	224	90

Figura 3.3: Instancia C101.txt de Solomon [56]

Los grupos de instancias difieren entre sí respecto al ancho de las ventanas de tiempo, medido entre la diferencia del fin e inicio de la ventana, algunos clientes tienen ventanas de tiempo muy estrecho, mientras que otros tienen ventanas de tiempo amplias. Presentan diferencias de igual forma en la ubicación de los clientes según sus coordenadas y densidad de las ventanas [26].

Con los datos de las instancias se construye una estructura de datos en el programa C++. Programa que se utilizará para la programación del algoritmo. Definimos: id=número de cliente, x=XCOORD, y= YCOORD, d=DEMAND, a=READYTIME,

b=DUE DATE y Ts=SERVICETIME. C++ es utilizado por ser un software que trabaja a nivel de procesador y los tiempos computacionales al resolver un problema son menores.

Se construye una matriz de distancias entre cliente y cliente, los datos iniciales proporcionan las coordenadas (x,y) de donde se encuentran situados los clientes, se supone que los vehículos viajan con una velocidad igual a 1.

Por lo tanto, el tiempo de viaje es igual a la distancia de viaje, utilizando la fórmula de la distancia euclidiana, que es la distancia ordinaria, es similar como si se pudiera medir con una regla, se calcula por medio del teorema de Pitágoras, tal como se muestra en la Ecuación 3.1.

$$d_{i,j} = \sqrt{(X_j - X_i)^2 + (Y_j - Y_i)^2} \tag{3.1}$$

De acuerdo a la metodología del *GRASP* representada en la Figura 3.4 es necesario aplicar dos fases: Construir soluciones y después mejorarlas, aplicando una búsqueda local, de acuerdo a la función objetivo, que en el caso de este problema será minimizar la distancia total del recorrido de las rutas. Se irá guardando la mejor solución hasta encontrar un criterio de parada, que será establecido en el programa controlado por el número de iteraciones.

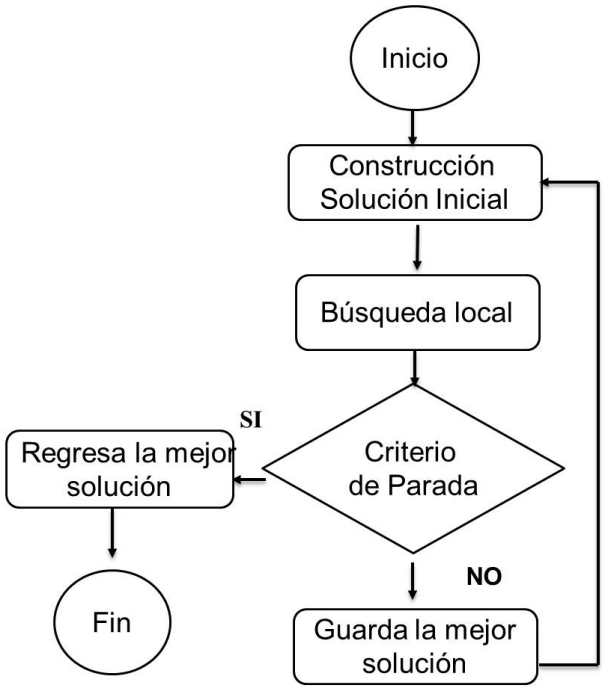


Figura 3.4: Diagrama de flujo de la metaheurística *GRASP*

3.2.2. Fase de construcción

Tal como se describió en la literatura, en la parte final de la Sección 2.3.1, el procedimiento *GRASP* en su fase de construcción inicia con definir la lista de candidatos (LC), por lo que se definió el conjunto de clientes como la lista de candidatos, los cuales se seleccionan de las estructura generada al leer las instancias.

La lista se debe ordenar de tal forma que al inicio se sitúen los clientes que mejoren la solución de acuerdo a la función objetivo de minimizar distancia.

Las ventanas de tiempo dificultan el problema, y es por ese motivo que no se ha definido en la literatura una forma específica de ordenamiento que proporcione mejores resultados de acuerdo a la función objetivo. Por tal motivo, con el fin de buscar los mejores resultados en la fase de construcción se determinaron tres formas de ordenar, basados en la ventanas temporales, ordenadas por el tiempo de entrada a dicha ventana, considerando dar el servicio primero aquellos clientes que lo requieran en un tiempo más temprano. Por lo que se consideraron tres formar de ordenar que son las siguientes:

1. Tiempo de inicio del intervalo de tiempo, a, de menor a mayor. Ejemplificado en la Fig. 3.5.

2. Tiempo de inicio del intervalo de tiempo, a, de mayor a menor. Ejemplificado en

la Fig. 3.6.

3. De acuerdo al valor del intervalo de salida, b, del cliente anterior en la lista sea menor que el valor de inicio del intervalo, a del cliente actual. Ejemplificado en la Fig. 3.7.

id	x	y	d	a	b	Ts
0	40	50	0	0	1236	0
20	30	50	10	10	73	90
5	42	65	10	15	67	90
13	22	75	30	30	92	90

Figura 3.5: Ejemplo de ordenamiento por a, de menor a mayor

id	x	y	d	a	b	Ts
0	40	50	0	0	1236	0
21	30	52	20	914	965	90
1	45	68	10	912	967	90
2	45	70	30	825	870	90

Figura 3.6: Ejemplo de ordenamiento por a, mayor a menor

id	x	y	d	a	b	Ts
0	40	50	0	0	1236	0
3	42	66	10	65	146	90
5	42	65	10	15	67	90
13	22	75	30	30	92	90

Figura 3.7: Ejemplo de ordenamiento por b del cliente anterior menor que al tiempo a, del cliente actual

Enseguida se selecciona la lista de candidatos restringida (LRC), de acuerdo con la metodología $GRASP$, se define el parámetro α. Este parámetro toma valores entre 0 y 1, entre más cerca este del cero disminuye la lista de candidatos restringida permitiendo que el algoritmo sea más voraz (greedy) al estar los datos ordenados de tal forma que favorezcan la función objetivo. Para calcular la LRC se aplica la ecuación 3.2.

$$LRC = LC^{*}\alpha \tag{3.2}$$

En este rubro se varía el parámetro α para la aplicación del algoritmo. Las instancias que se utilizaron varían de 25, 50, y 100 clientes, quienes estarían formando el conjunto de LC. Tomando en cuenta la experiencia de los investigadores que han estudiado el $VRPTW$ quienes aseguran resultados de calidad si la LRC se establece en 3 clientes [60], [61], se utilizó una LRC variable de acuerdo a la siguiente estrategia:

1. Mientras la LC sea menor del 50 % del total de los clientes, la LRC se especifica en 3 clientes. Es decir, con un α que es igual a 3 clientes dividido entre el total de clientes iniciales (n). De tal forma que si n es igual a 100, α será 0.03 y al aplicar la ecuación 3.2, la LRC será de 3 clientes.

2. Si la lista de clientes es mayor a 50 % y menor de 75 %, la LRC es de 2 clientes.

3. El resto, es decir cuando resta el 25 % de los clientes sin asignar a una ruta la lista restringida se reduce a un solo cliente.

Ya que se tiene definida la LRC, de forma aleatoria se selecciona un cliente, si es el primer cliente que se selecciona se conecta directamente al depósito, e inicia la primer ruta. A partir de ese momento se construye la estructura de resultados en donde se define el predecesor y sucesor de cada cliente, con el fin de ir construyendo las rutas que serán asignadas a cada vehículo.

Se inicia el cálculo de la distancia y el de tiempo acumulado de la ruta (T_{acum}), (Ecuación 3.3), al asignar el primer cliente, se considera el tiempo de recorrido del depósito (0) al cliente i, (t_{0i}) aumentando el tiempo de servicio Ts_i del cliente i.

$$T_{acum\_i} = t_{0i} + Ts_i \tag{3.3}$$

Se deberá de respetar las ventanas temporales, considerando que en el momento de dar el servicio a un cliente se toma en cuenta el tiempo de traslado entre depósito y cliente, o entre cliente y cliente y el tiempo de servicio del cliente. Es importante considerar el tiempo de espera en caso de que se llegue antes del inicio de la ventana de tiempo. De igual forma se acumula la demanda al ir conectando clientes a cada ruta para no exceder la capacidad permitida en el vehículo.

En la construcción de la solución se construirán las rutas en paralelo, es decir se irán construyendo varias rutas al mismo tiempo. Desde el segundo cliente y hasta el último de la lista se evaluarán las siguientes posibilidades para conectar al cliente a la ruta que nos arroje la menor distancia y cumpla con las restricciones de ventana de tiempo:

1. En caso de que haya una sola ruta en construcción, se calcula el tiempo de recorrer el último cliente i de la ruta, al cliente j seleccionado . Para que sea asignado a esa ruta deberá asegurarse de estar dentro de la ventana de tiempo del cliente seleccionado j, cumpliendo con la restricción de la Fórmula 3.4. Si cumple la restricción, el cliente j es conectado al final de la ruta al cliente i; en caso contrario

se conecta directamente al depósito dando inicio a una nueva ruta.

$$T_{acum\_i} + t_{ij} \leq b_j \tag{3.4}$$

Al programar la Fórmula 3.4, se asegura que el cliente es candidato para ser asignado a la ruta, es decir, cumple la restricción de la ventana de tiempo, sin embargo se consideran dos circunstancias para el cálculo del tiempo acumulado:

a) Si el cliente llega después del inicio de la ruta, restricción que se verifica con la Fórmula 3.5.

$$T_{acum\_i} + t_{ij} \geqslant a_j \tag{3.5}$$

Se aplica la Fórmula 3.6

$$T_{acum\_j} = T_{acum\_i} + t_{ij} + Ts_j \tag{3.6}$$

b) En caso de que se llegue antes del inicio de la ventana de tiempo, se verifica con la Fórmula 3.7.

$$T_{acum\_i} + t_{ij} \leq a_j \tag{3.7}$$

Entonces se deberá considerar el tiempo de espera y se hace el cálculo del tiempo acumulado con la Fórmula 3.8.

$$T_{acum\_j} = a_j + Ts_j \tag{3.8}$$

2. En caso de existir dos rutas o más, al seleccionar un nuevo cliente y hasta que la lista de candidatos quede en cero, se comparará primeramente el tiempo de trasladarse de los clientes que estén al final de las ruta ya construidas al cliente que se haya seleccionado, conectando el seleccionado a la ruta que proporcione el mínimo tiempo de trasladarse y que cumpla con las ventanas de tiempo, tal como se describió anteriormente.

Ejemplificando gráficamente la construcción de la ruta, se observa en al Figura 3.8, que están construidas tres rutas. Suponiendo que el cliente seleccionado es el 7, se evalúa la posibilidad de conectarlo al cliente 6, 5 y 3. Se selecciona el que arroje menor distancia (tiempo de traslado), por observación se concluye que conectarlo al cliente 3 nos daría la menor distancia.

Sin embargo antes de asignar el clientes 7 como sucesor del cliente 3, se debe de evaluar si cumple con la ventana de tiempo, es decir, que el tiempo que se lleve acumulado en la ruta (cliente 3) más el tiempo de recorrer del cliente 3 al cliente 7 permita entrar a tiempo en el final de la ventana del cliente 7. En caso contrario se evaluaría después el cliente 6, por ser el que sigue en distancia y finalmente el 5.

En caso de no cumplir con las restricciones de tiempo, de las tres rutas, se conecta al depósito generando una nueva ruta.

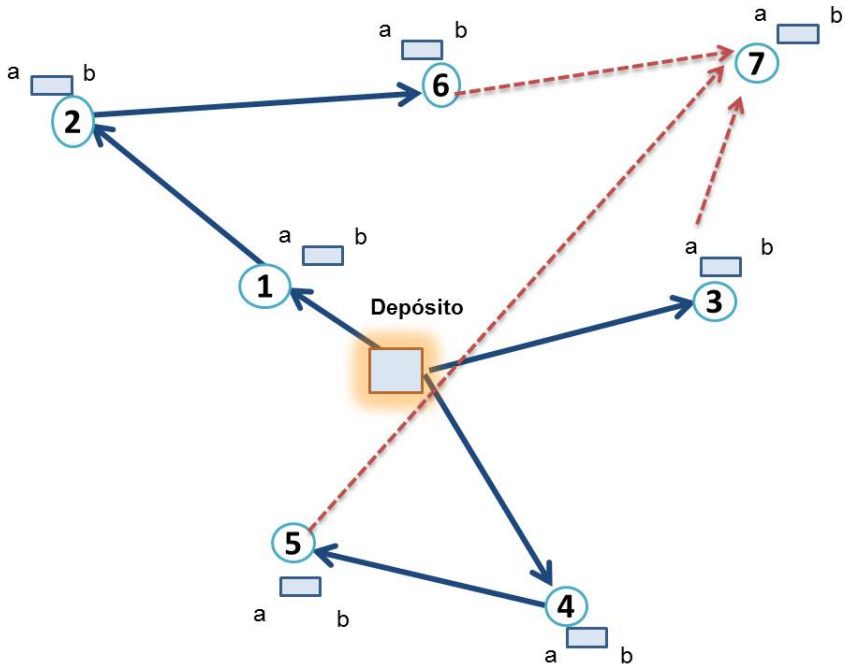


Figura 3.8: Ejemplo de la construcción de una ruta del *VRPTW*

3.2.3. Fase de mejora - búsqueda local

Una vez que se tiene la ruta construida se continua con la segunda fase del algoritmo *GRASP*, aplicar una búsqueda local. En la programación de éste algoritmo se aplican los siguientes procedimientos de mejora:

1. Se hace un recorrido en todos los vértices, detectando aquellos clientes que quedaron solos conectados al depósito y los reubica en otra ruta, siempre que cumpla con las restricciones de demanda, de ventanas de tiempo, y disminuya la distancia total del recorrido. Representado gráficamente, en la Figura 3.9 se observa en el inciso (a), que el cliente 3, en la ruta de construcción quedó solo conectado al depósito, una vez que se evalúa que cumpla con las restricciones de demanda y ventanas de tiempo, queda conectado en la ruta 2, tal como se observa en el inciso (b).

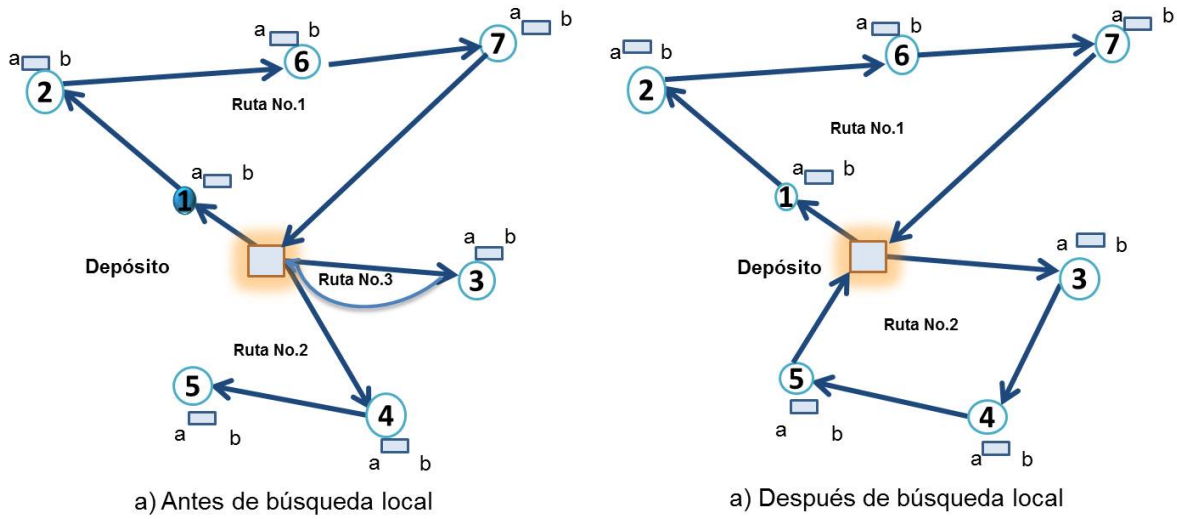


Figura 3.9: Ejemplo de la aplicación de búsqueda local destruyendo ruta con un solo cliente

2. La segunda mejora programada en el algoritmo hace un recorrido por cada vértice y evalúa si disminuye la distancia al reubicarlo al final de la ruta, en caso de cumplir con la disminución de la distancia y cumplir con las restricciones del *VRPTW*, realiza el cambio, en caso contrario deja la ruta tal como se construyó. En la Figura 3.10 se observa en la ruta 1, que el cliente tres se encuentra más cerca del depósito que el cliente 7, por lo que evaluando que cumpla con las restricciones propias del *VRPTW*, se reubica tal como observamos en la parte (b) de la Figura 3.10, este cambio se lleva siempre que minimice la función objetivo.

3. La búsqueda local tres incluida en la programación del algoritmo, aplicado en la solución de problemas *VRPTW* radica en la destrucción de una ruta, reubicando cada uno de los vértices en otra ruta donde cumpla con las restricciones, logrando minimizar la función objetivo. En la Figura 3.11 se muestra un ejemplo, donde se elimina una ruta.

4. Finalmente, se programó como búsqueda local, aplicar la metodología que se utiliza para resolver el problema del agente viajero a cada ruta, y así obtener soluciones de mejor calidad, de acuerdo a la función objetivo de minimizar distancia. En la Figura 3.12 se observa diferentes rutas que se pueden construir y como se mejora al aplicar esta búsqueda.

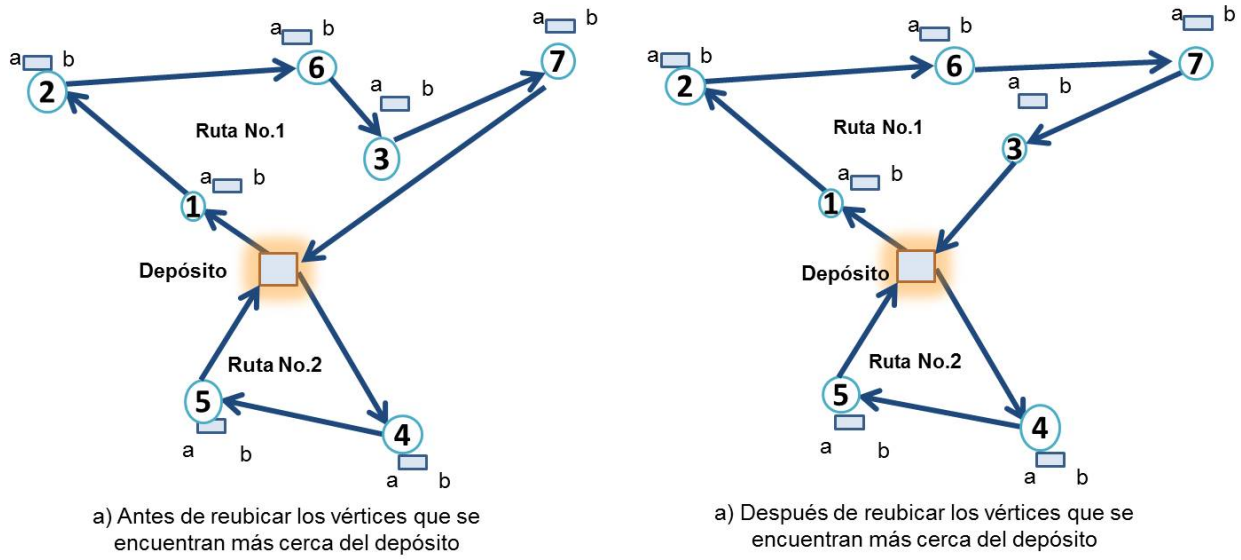


Figura 3.10: Ejemplo de la aplicación de búsqueda local reubicando los vértices más cercanos al depósito

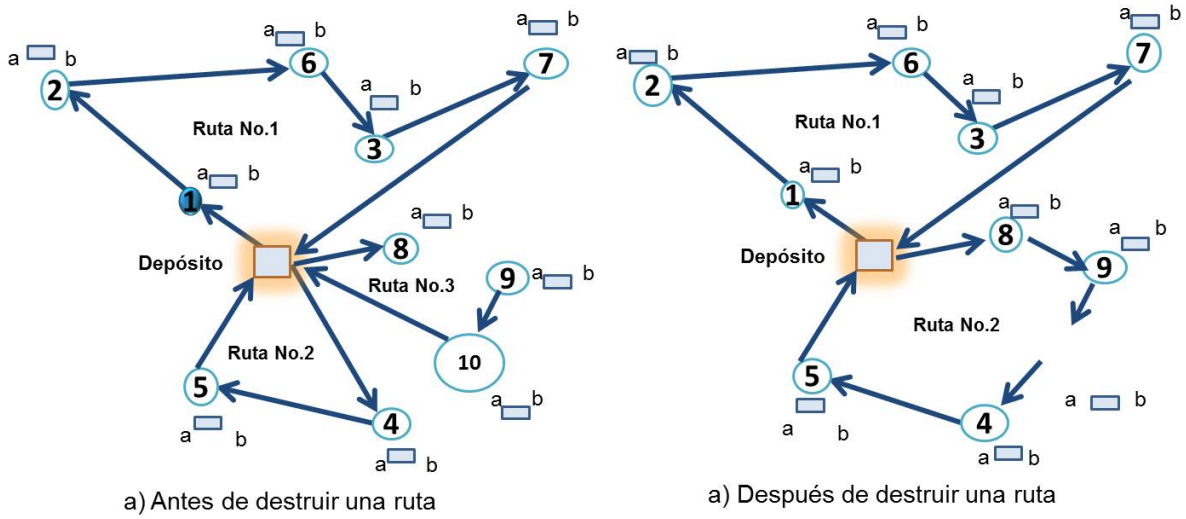


Figura 3.11: Ejemplo de la aplicación de búsqueda local destruyendo una ruta

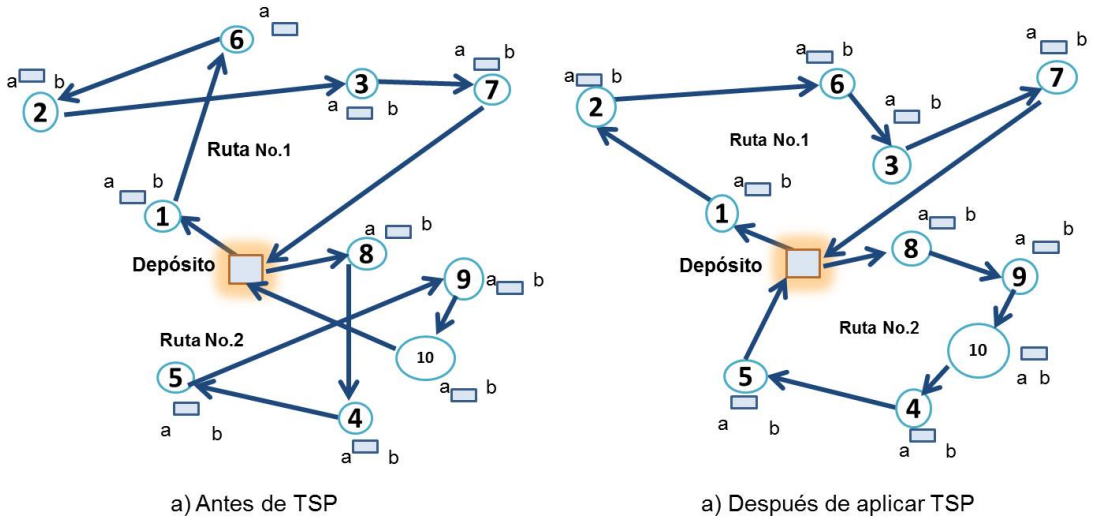


Figura 3.12: Ejemplo de la aplicación de búsqueda local con metodología que se utiliza para resolver el agente viajero

En el siguiente capítulo se muestran los resultados obtenidos con los parámetros especificados en la Sección 3.2.2, utilizando las tres formas de ordenamiento y el valor de α variable ya establecido, esta etapa de la investigación se identificará como **Fase 1**. Sin embargo, como se explica en el Capítulo 4, los resultados obtenidos no fueron satisfactorios comparados con los resultados conocidos de Solomon, por lo que se aplicaron otras opciones de ordenamiento y diferentes valores de α, realizando varias pruebas piloto, seleccionando aquellas que arrojaron las mejores soluciones de acuerdo a la función objetivo.

Con los parámetros que arrojaron soluciones de mejor calidad se realizó un diseño de experimento (DOE), utilizando el método Taguchi, con el fin de establecer los parámetros recomendados para la fase de construcción en el algoritmo basado en la metaheurística *GRASP* para la solución de problemas de *VRPTW*, al que solo por simplificar, se le identificará como *G-VRPTW*.

Lo anterior se basa en que los algoritmos metaheurísticos son procedimientos de tipo caja negra que analizan un subconjunto de soluciones posibles para resolver un problema o un conjunto de instancias. Sin embargo, previo a su ejecución se requiere seleccionar un vector de parámetros, el cuál afecta en gran medida la eficiencia de la metaheurística al ser utilizada para resolver un problema determinado.

Al procedimiento de seleccionar el vector de parámetros se le conoce como calibración. Los parámetros se clasifican en cualitativos y cuantitativos. Los primeros se relacionan con procedimientos (ejemplo: clasificación y procedimiento para seleccionar clientes),

mientras que los cuantitativos se asocian a valores específicos (ejemplo: número de iteraciones y número de clientes).

Para obtener un buen rendimiento en un algoritmo metaheurístico al resolver un problema, es necesario considerar en una etapa previa a su uso, un proceso para la selección del vector de parámetros, que en la mayoría de los casos se reduce a un proceso de exploración, donde el vector de parámetros es seleccionado con base en la experiencia, *calibración manual* [72] o, realizando una exploración en la literatura en problemas similares, es decir *calibración por analogía* [73].

Los resultados son mostrados en el capítulo siguiente, en donde se analiza el efecto de los dos procedimientos para la selección del vector de parámetros óptimo P^* en *G-VRPTW*, en las soluciones que éste genera.

Capítulo 4

Pruebas y Resultados

En este capítulo se presentan los resultados obtenidos en la modelación y programación del algoritmo en C++. Se detallan y analizan los resultados obtenidos antes de aplicar las búsquedas locales, se presentan de igual forma los resultados obtenidos al aplicar el método Taguchi.

Los resultados presentados en este proyecto se obtuvieron utilizando una computadora personal con Intel Core i7-5500 CPU at 2.4 GHz.

La calidad de la solución se mide en términos distancia mínima de la ruta, esto es, la distancia recorrida por un vehículo visitando a todos los clientes, partiendo y regresando al depósito. Se consideró en el presente trabajo que una unidad de distancia recorrida es una unidad de tiempo.

4.1. Resultados del *G-VRPTW* en etapa de construcción, Fase 1

Los resultados obtenidos en la fase de construcción son resultado de un promedio de diez ejecuciones del algoritmo en ocho instancias por grupo seleccionadas al azar de las presentadas por Solomon en 1987 [56].

Se estableció como criterio de parada del algoritmo 50,000 iteraciones. En la Tabla 4.1 se muestran los resultados obtenidos con los diferentes parámetros de ordenamientos mencionados en la sección 3.2.2, donde:

- Instancias. Es la nomenclatura de las instancias del grupo de Solomon.
- Óptimo. Valor óptimo mejor conocido según la literatura de las instancias de Solomon.
- Prom. Indica el promedio obtenido de las diez corridas de las ocho instancias utilizadas.

- GAP. Es el indice que mide que tan buena es la solución obtenida contra la mejor conocida de la instancia. Explicado a detalle en la Figura 3.1
- PGRAL. Es el promedio general del % Gap de los resultados que se obtuvieron.

Tabla 4.1: Resultados obtenidos en la fase construcción con los diferentes parámetros de ordenamiento

Primer Ordenamiento									
	25 clientes			50 clientes			100 clientes		
Instancias	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP
C1	190.50	377.04	97.92	361.60	814.70	125.31	826.62	1990.68	140.82
C2	214.45	392.27	82.92	357.50	864.36	141.78	588.00	1689.93	187.40
R1	481.65	763.62	58.54	796.86	1447.06	81.59	1266.43	2408.85	90.21
R2	384.67	781.39	103.13	654.10	1476.06	125.66	965.88	2646.33	173.98
RC1	359.25	633.25	76.27	731.32	1384.85	89.36	1396.45	2767.05	98.15
RC2	319.27	719.06	125.22	608.75	1565.38	157.15	1119.23	2997.89	167.85
PGRAL.	324.96	611.08	90.66	610.45	1258.74	120.14	1027.10	2416.79	143.06
Segundo Ordenamiento									
	25 clientes			50 clientes			100 clientes		
Instancias	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP
C1	190.50	683.80	258.95	361.60	1655.86	357.93	826.62	4149.64	402.00
C2	214.45	539.92	151.77	357.50	1355.56	279.18	588.00	3532.69	500.80
R1	481.65	854.80	77.47	796.86	1782.90	123.74	1266.43	3253.99	156.94
R2	384.67	760.60	97.73	654.10	1521.98	132.68	965.88	2632.02	172.50
RC1	359.25	836.790	132.93	731.32	1896.95	159.39	1396.45	3458.29	147.65
RC2	319.27	788.21	146.88	608.75	1896.42	211.53	1119.23	3414.01	205.03
PGRAL.	324.96	744.02	144.28	585.02	1684.95	210.74	1027.10	3406.77	264.15
Tercer Ordenamiento									
	25 clientes			50 clientes			100 clientes		
Instancias	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP
C1	190.50	379.00	98.95	361.60	833.20	130.42	826.62	2050.74	148.09
C2	214.45	375.00	74.87	357.50	927.98	159.58	588.00	1826.06	210.55
R1	481.65	731.67	51.91	796.86	1361.42	70.85	1266.43	2281.37	80.14
R2	384.675	735.34	91.16	654.10	1406.22	114.99	965.88	2432.42	151.83
RC1	359.25	585.70	63.04	731.32	1240.92	69.68	1396.45	2584.14	85.05
RC2	319.27	680.34	113.09	608.75	1428.62	134.68	1119.23	2867.27	156.18
PGRAL.	324.96	581.17	82.16	585.02	1199.72	113.36	1027.10	2340.33	138.64

Concentrando la información por número de clientes, se observa en las Figuras 4.1, 4.2 y 4.3, que el ordenamiento tres, ordenado por el valor de entrada de la ventana de tiempo actual menor que el valor de salida de la ventana de tiempo del cliente anterior es el que arroja los mejores resultados en la fase de construcción. Se observa de igual forma que el ordenamiento dos, que consiste en insertar en la ruta los clientes que solicitan el servicio más temprano, es el que arroja los resultados más altos en todos los grupos de instancias.

Sin embargo comparando los resultados obtenidos en las tres formas de ordenamiento, con los resultados del mejor conocido hasta este momento en las instancias de Solomon, se aprecia que se encuentran muy alejados. En la sección siguiente se presentan los resultados obtenidos aplicando las búsquedas locales descritas en la Sección 3.2.3.

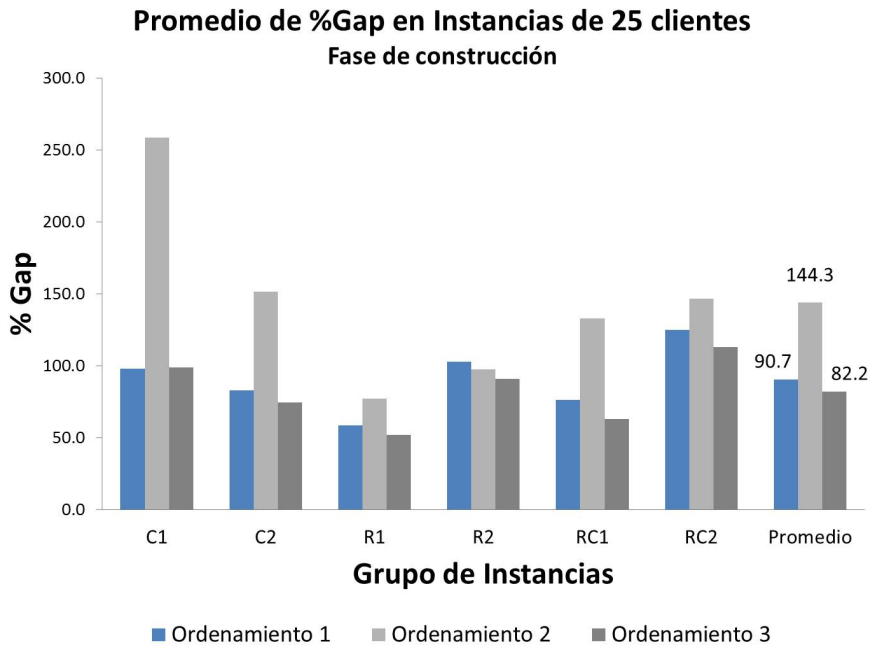


Figura 4.1: Resultado obtenidos en la fase de construcción de instancias de 25 clientes

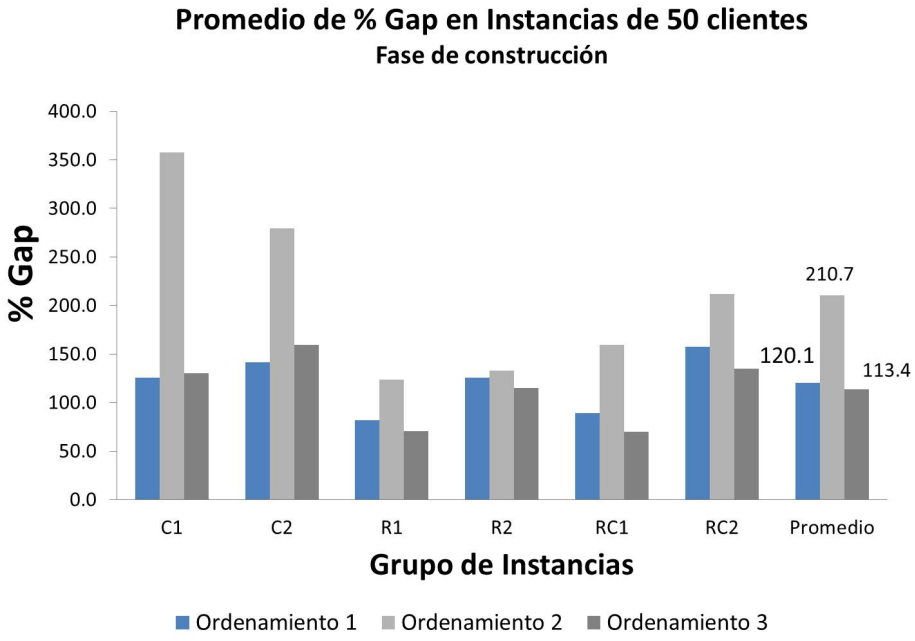


Figura 4.2: Resultado obtenidos en la fase de construcción de instancias de 50 clientes

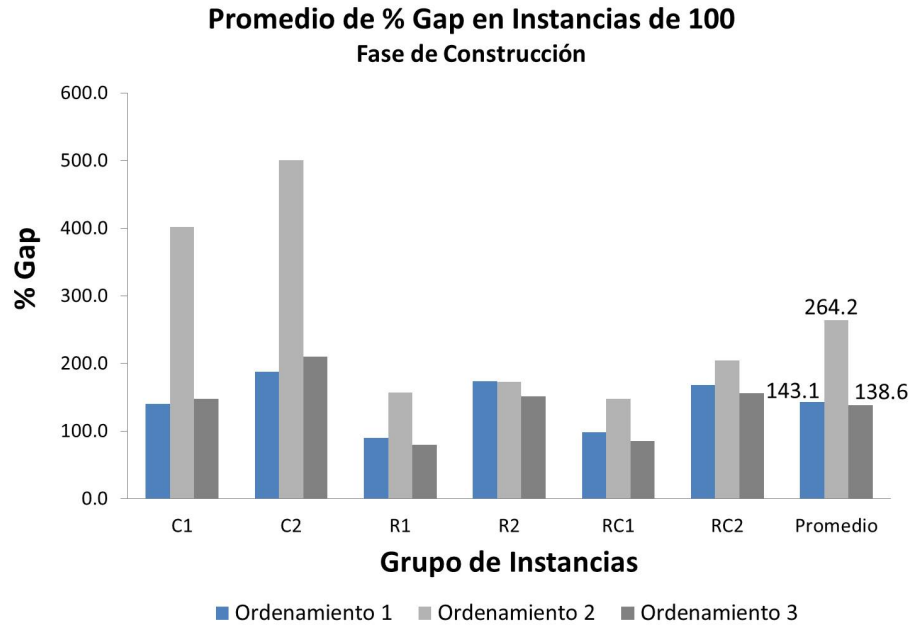


Figura 4.3: Resultado obtenidos en la fase de construcción de instancias de 100 clientes

4.2. Resultados con búsqueda local en la Fase 1

Una vez aplicadas las búsquedas locales descritas anteriormente en la Sección 3.2.3 se muestran los resultados obtenidos en la Tabla 4.2. Destacando los resultados obtenidos en el grupo de instancias RC2 del conjunto de 25 clientes, donde se refleja en el segundo ordenamiento un resultado negativo en el % Gap, es decir se obtuvo un resultado por abajo del resultado mejor conocido en la literatura. En el conjunto de instancias de 50 clientes los resultados más bajos se reflejan en las instancias C1, R1 y RC1, por lo que el *G-VRPTW* con los parámetros establecidos en la **Fase 1** funciona muy bien para las instancias con ventanas de tiempo estrechas y horizonte de programación corto según las características de las instancias detalladas en la Tabla 3.1.

En las Figuras 4.4, 4.5 y 4.6, se observan los resultados agrupados por número de clientes. Apreciando que en instancias de 25 clientes se obtiene un Gap promedio de 5 %, en instancias de 50 clientes el resultado más bajo es 16. 4 % presentado con el ordenamiento tres y finalmente en 100 clientes se obtiene un Gap promedio del 29.3 % en el ordenamiento uno. El conjunto de instancias R1 son las que se ajustaron mejor a los parámetros establecidos en el algoritmo en sus tres fases de ordenamiento.

Tabla 4.2: Resultados obtenidos en la fase de mejora, por grupo de instancias y parámetros de ordenamiento

Primer Ordenamiento

25 clientes				50 clientes			100 clientes		
Instancias	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP	Óptimo	Prom	GAP
C1	190.50	193.20	1.42	361.60	392.76	8.62	826.62	1039.94	25.81
C2	214.45	235.96	10.03	357.50	439.92	23.05	588.00	782.42	33.07
R1	481.65	493.85	2.53	796.86	854.00	7.17	1266.43	1434.62	13.28
R2	384.67	423.76	10.16	654.10	813.57	24.38	965.88	1299.54	34.54
RC1	359.25	359.66	0.12	731.32	786.91	7.60	1396.45	1842.59	31.95
RC2	319.27	338.74	6.10	608.75	778.86	27.94	1119.23	1532.71	36.94
PGRAL.	324.96	340.86	5.06	585.02	677.67	16.46	1027.10	1321.97	29.26

Segundo Ordenamiento

25 clientes				50 clientes			100 clientes		
Instancias	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP
C1	190.50	201.72	5.89	361.60	444.51	22.93	826.62	1243.50	50.43
C2	214.45	239.39	11.63	357.50	431.03	20.57	588.00	980.98	66.83
R1	481.65	498.90	3.58	796.86	863.65	8.38	1266.43	1552.99	22.63
R2	384.67	398.83	3.68	654.10	724.06	10.70	965.88	1214.69	25.76
RC1	359.25	384.51	7.03	731.32	906.50	23.95	1396.45	1854.23	32.78
RC2	319.27	317.71	**-0.49**	608.75	778.86	27.94	1119.23	1532.71	36.94
PGRAL.	324.96	340.18	5.22	585.02	691.43	19.07	1027.10	1372.33	39.22

Tercer Ordenamiento

25 clientes				50 clientes			100 clientes		
Instancias	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP	Óptimo	Prom.	GAP
C1	190.50	193.01	1.32	361.60	393.30	8.77	826.62	1133.33	37.10
C2	214.45	230.93	7.69	357.50	457.55	27.98	588.00	914.52	55.53
R1	481.65	497.05	3.20	796.86	879.33	10.35	1266.43	1554.61	22.75
R2	384.67	415.58	8.04	654.10	764.07	16.81	965.88	1361.82	40.99
RC1	359.25	365.70	1.78	731.32	833.37	13.95	1396.45	1737.86	24.45
RC2	319.275	344.632	7.94	608.75	732.22	20.28	1119.233	1591.672	42.21
PGRAL.	324.967	341.14	4.995	585.024	676.641	16.358	1027.107	1372.333	37.173

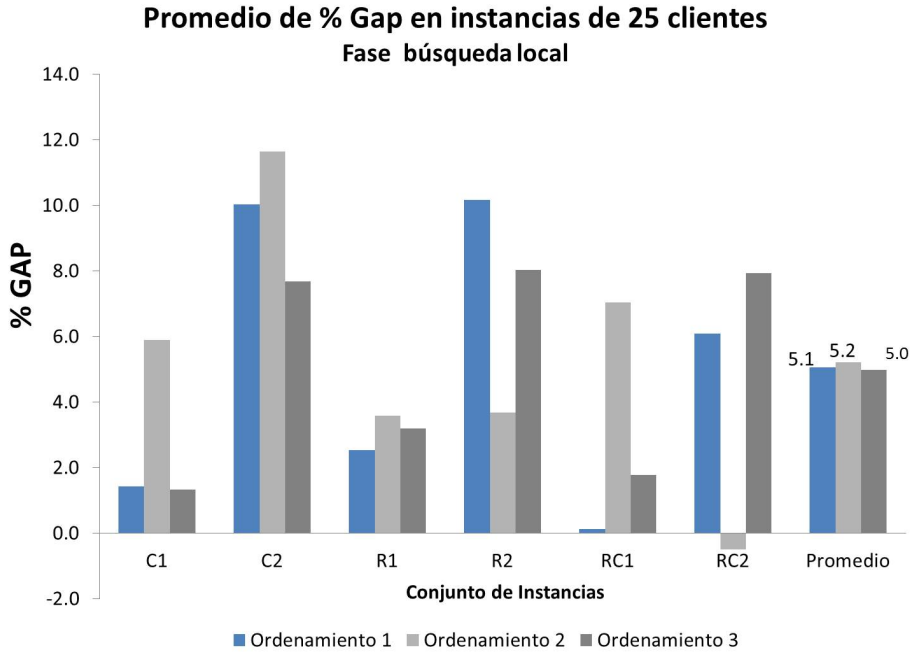


Figura 4.4: Resultado obtenidos en la fase de mejora de instancias de 25 clientes

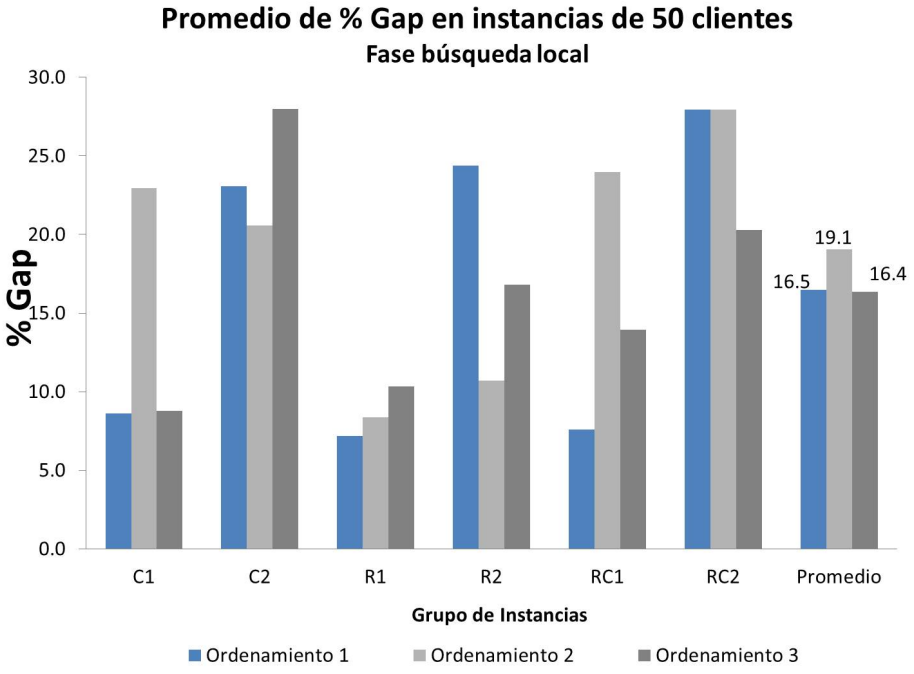


Figura 4.5: Resultado obtenidos en la fase de mejora de instancias de 50 clientes

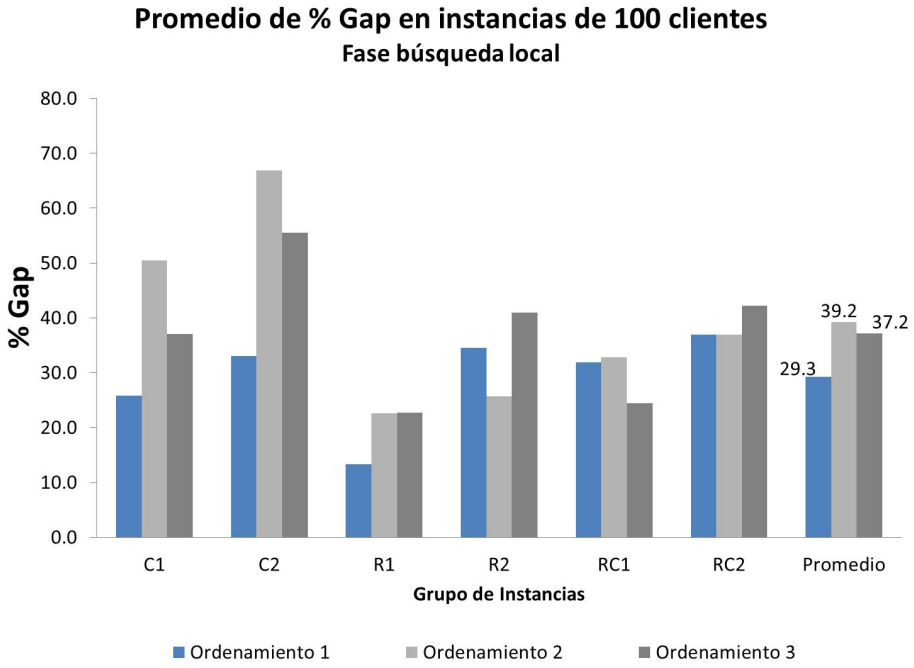


Figura 4.6: Resultado obtenidos en la fase de mejora de instancias de 100 clientes

Los resultados en el grupo de instancias de 100 clientes es evidente que se tienen que mejorar ya que como se observa en la Figura 4.6, el % Gap en promedio esta muy alejado de la mejor solución reportada en la literatura consultada. En la Figura 4.7 se aprecia un concentrado de resultados con búsqueda local (CB) y sin búsqueda local (SB). Las soluciones obtenidas al aplicar las búsquedas locales que se modelaron lograron mejorar los resultados de la fase de construcción, tal como se aprecia en la Tabla 4.3, en promedio, el primer ordenamiento es el que proporciona los mejores resultados, con 16.95 % Gap.

Tabla 4.3: Resultados obtenido en la fase de construcción y fase de búsqueda local

	25 clientes		50 clientes		100 clientes		
Ordenamiento	SB	CB	SB	CB	SB	CB	Prom.
1	90.7	5.1	120.1	16.4	143.10	29.3	16.95
2	144.3	5.2	210.7	19.1	264.2	39.2	21.17
3	82.2	5	113.4	16.4	138.6	37.2	19.53

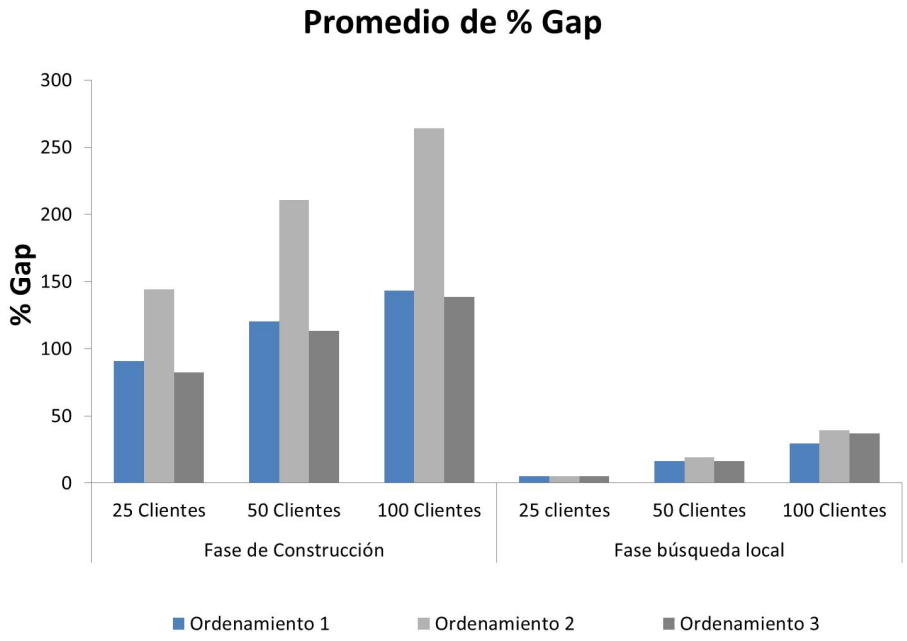


Figura 4.7: Comparación de los resultados obtenidos en la fase de construcción y con búsqueda local por cada ordenamiento

En general, se aprecia poca variación en los diferentes tipos de ordenamiento en los resultados obtenidos al aplicar búsqueda local, como se aprecia en la Figura 4.7. Se observa de igual forma en la Figura 4.7 que las búsquedas locales aplicadas al *G-VRPTW* mejoran las soluciones.

En la Figura 4.8 se detallan los resultados por grupo de instancias del ordenamiento uno, que en promedio arroja los resultado más cercanos al conocido en la literatura. La Tabla 4.4 indica que de las 144 instancias de prueba en 63 de ellas se logró un % Gap por abajo del 5 %, 22 instancias se comportaron abajo del 10 % de Gap; mientras que el resto de la instancias reportan resultados arriba del 10 %.

Tabla 4.4: Resultados de % Gap por instancia de prueba utilizada en **Fase 1**

% Gap	25 clientes	50 clientes	100 clientes
0 - 5 %	33	8	2
5-10 %	10	10	2
10-20 %	5	11	6
20-30 %	0	14	13
30-40 %	0	5	11
40-50 %	0	0	11
50-60 %	0	0	2
60-70 %	0	0	0
70-80 %	0	0	1
Total	**48**	**48**	**48**

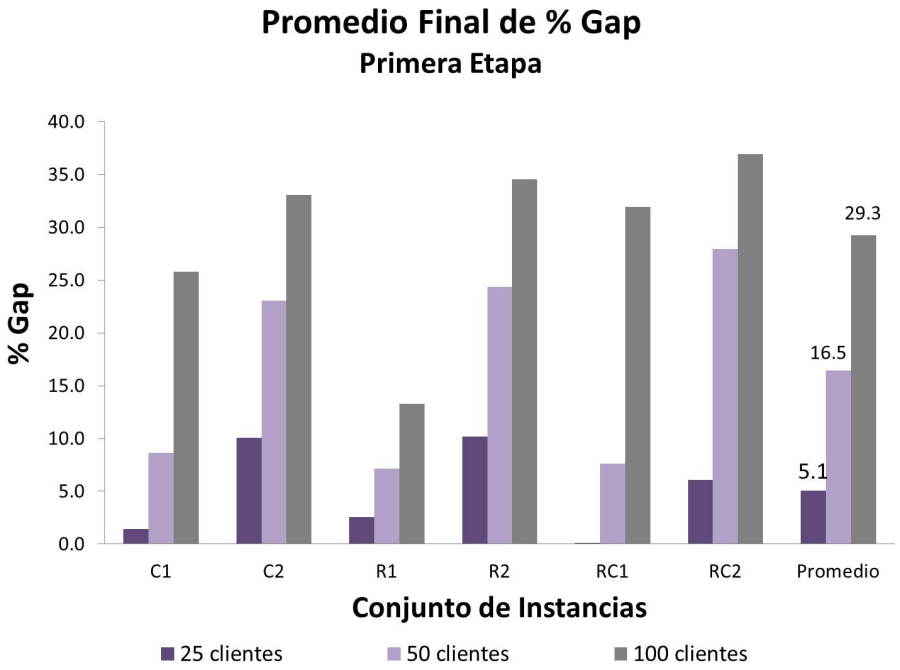


Figura 4.8: Resultados obtenidos por instancias con el mejor ordenamiento

4.3. Diseño de experimentos

Con el fin de mejorar y establecer los parámetros del *G-VRPTW* se realizó un diseño de experimentos. Utilizando el concepto de arreglos ortogonales de Taguchi, ya que el número de experimentos puede ser significativamente reducido, y por lo tanto, el tiempo y el costo causado por múltiples experimentos también se reducen [74].

El diseño experimental de Taguchi tiene como objetivo minimizar la variabilidad en la operación, seleccionando las combinaciones más adecuadas de los niveles de factores controlables en comparación con los factores incontrolables que crean variabilidad para un producto u operación. El diseño de parámetros de Taguchi o la metodología de diseño robusto implica la maximización del rendimiento y la calidad a un menor costo [75].

Otro componente importante del método de Taguchi es la señal a ruido S/R, que es una escala de medición que se ha utilizado en la industria de la comunicación durante casi un siglo. Taguchi la generaliza y la utiliza como un factor de calidad, entre más alta la señal a ruido indica un proceso de mejor calidad [76].

En el método de Taguchi, las características de calidad se presentan en tres escenarios: Lo más bajo mejor, lo nominal es mejor y entre más alto mejor. En el caso de este trabajo el objetivo es minimizar la distancia, por lo que utilizamos lo más bajo es mejor, aplicando la siguiente ecuación:

$$S/R = -10Log(1/n \sum_{i=1}^{n} Y_i^2)$$

Donde:

S/R es la razón señal a ruido

n es el número de configuraciones

i es la configuación

Y_i es la media de los datos de la configuración i

4.3.1. Aplicación del método Taguchi

En la aplicación correcta del método Taguchi se deben de cumplir con los siguientes pasos [77]:

1. **Formular el problema**. El objetivo del *G-VRPTW* es minimizar la distancia, por lo que se requiere establecer los parámetros que afectan la solución final y mejorar los resultados obtenidos mostrados en la Figura 4.8.

2. **Identificar las características de rendimiento de salida más relevantes para el problema**. En este rubro se consideraron los resultados finales concentrados en la Tabla 4.2. El % Gap es la métrica que señala que tan alejados están los datos del mejor conocido, como ya se ha mencionado en la Sección 3.1, por lo que con base en los resultados obtenidos en la **Fase 1**, se torna evidente la necesidad de calibrar los parámetros que afectan la solución en busca de mejores resultados.

3. **Identificar los factores de control. Factores de señal a ruido (si los hay)**. Los factores de control que se determinan son aquellos que afectan en la solución. La metodología del *GRASP* presenta 2 parámetros a medir que son el orden de los clientes y el valor de α, un tercer parámetro a controlar en el *G-VRPTW* es el número de iteraciones. En lo que respecta a los factores de ruido que son aquellos factores no controlados que afecten a los resultados, no se consideró ninguno ya que el objetivo final al aplicar el método Taguchi es determinar un nivel que mejore los resultados de los factores controlables.

4. **Seleccionar los niveles del factor.** Los niveles que se establecieron para los parámetros o factores que influyen en la solución final del algoritmo, se pueden consultar en la Tabla 4.5 y fueron elegidos bajo los siguientes criterios:

   a) Ordenamiento. EL orden es la forma en que se inician los datos en la fase de construcción para iniciar las rutas. Se definieron los siguientes niveles:

   - **Nivel 1**. Iniciar con el cliente que requiere ser atendido más temprano según su ventana de tiempo, se identificará por: [a]. Se construyen rutas considerando el tiempo.

   - **Nivel 2**. Iniciar con el cliente que esté más cerca del depósito [dd]. Se construyen rutas considerando la distancia o tiempo de recorrido.

   - **Nivel 3**. Mixto. En éste ordenamiento se hizo una combinación de los dos anteriores, considerando: el primer 20 % de clientes con el Nivel 2, el siguiente 60 % con el Nivel 1 y el 20 % restante con el Nivel 2. Con el fin de iniciar y terminar la construcción de las rutas con los clientes más cercanos al depósito y en la fase intermedia de la construcción de las rutas ir considerando a los clientes que solicitan el servicio más temprano.

   b) Valor de α. Factor que determina que tan voraz es la metaheurística *G-VRPTW*. Los niveles son:

   - **Nivel 1 = Variable**. Se considera el valor de α variable durante la ejecución del algoritmo, esto es el 50 % de la *LRC* es definida en 3 clientes, es decir, la selección aleatoria del candidato que formará parte de la ruta de solución es seleccionado en 3 clientes, tal como se explica en la Sección 3.2.1, lo anterior deja el valor de α en función del número de clientes, aplicando la Ecuación 3.2. Del 50 al 75 % selecciona en 2 clientes y el 25 % restante hace la selección en un cliente.
   - **Nivel 2 = 0.90**. Al aumentar el valor de α se está introduciendo variabilidad al momento de seleccionar aleatoriamente al cliente, ya que la *LRC* será mayor y esto ayudará al proceso para evitar caer en óptimos locales.

- **Nivel 3 = 3 clientes**. En este nivel no se especifica un valor de α fijo, si no que está en función del tamaño de la población al determinarlo en 3 clientes, considerando la Fórmula 3.2 y basados en la revisión de la literatura [61], [60].

c) Iteraciones. El número de veces que se iterará el algoritmo buscando mejorar la solución que va guardando en memoria, estableciendo los siguientes niveles:

- **50,000**
- **100,000**
- **150,000**

El número de iteraciones se establecieron en los niveles anteriores basados en las diversas pruebas pilotos, considerando incrementos en igual magnitud. En la Tabla 4.5, se muestra el concentrado de los factores con los niveles.

Tabla 4.5: Tabla de factores seleccionados

Factor	Nivel 1	Nivel 2	Nivel 3
Ordenamiento	a de menor a mayor	cliente más cerca al depósito al más alejado	mixto
Alfa	Variable	0.90	3 clientes
Iteraciones	50000	100000	150000

5. **Diseñar el arreglo ortogonal adecuado al problema**. Los arreglos ortogonales propuestos por Taguchi son diseños que tienen la propiedad de ortogonalidad, misma que también poseen los diseños factoriales. Estos arreglos son diseños factoriales completos, fraccionados o mixtos, dependiendo del número de factores a estudiar en un caso particular. Taguchi desarrolló una serie de arreglos que denominó:

$$L_a(b)^C$$

Donde:

a = Representa el número de pruebas o condiciones experimentales que se deben realizar. Esto es, el número de renglones o líneas en el arreglo.

b = Representa los diferentes niveles a los que se tomará cada factor.

c = Es el número de efectos independientes que se pueden analizar, esto es, el número de columnas en el arreglo.

Se seleccionó un arreglo ortogonal $L_9 3^{4-2}$ de los propuestos por Taguchi que permite estudiar máximo cuatro factores a tres niveles cada uno, como se muestra en la Tabla 4.6 [78]. De acuerdo a los factores y niveles definidos en el experimento del presente trabajo, ver Tabla 4.5, solo se ocupan las primeras 3 columnas,

correspondientes al factor ordenamiento, factor α y factor iteraciones, respectivamente, quedando así, un arreglo ortogonal $L_9 3^3$.

Tabla 4.6: Arreglo L_9 propuesto por Taguchi

Arreglo $L_9 3^{(4-2)}$

Número de corrida	Número de columna			
	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1
2 factores: columnas 1, 2. 3 factores: columnas 1, 2, 3. 4 factores: columnas 1, 2, 3, 4.				

6. **Ejecutar el experimento y recopilar los datos necesarios**. Se ejecutó el programa de acuerdo al arreglo ortogonal Taguchi considerando el 20 % de las instancias seleccionadas aleatoriamente, se realizaron 10 replicas por cada uno de los arreglos, tal como se muestra en la Tabla 4.7, utilizando el software minitab 17. El concentrado de los resultados al ejecutar el *G-VRPTW* en el 20 % de las instancias se puede observar en la Tabla 4.8, en donde se observan en las configuraciones 2, 5, 8 y 9 valores negativos en el grupo de instancias de 25 clientes R1 y RC1, identificadas como clientes que se encuentran distribuidos aleatoriamente y en grupo, con ventanas estrechas y corto horizonte de programación. En las instancias de 50 clientes se observa que el grupo RC1 se mantiene en todas las configuraciones en un Gap menor al 10 %. En 100 clientes el grupo RC2 refleja los resultados más bajos.

Tabla 4.7: Concentrado de factores y niveles en la aplicación del método Taguchi

Config.	*Orden*	*N*	*Alfa*	*N*	*Iteraciones*	*N*
1	a<a	1	Variable	1	50,000	1
2	a<a	1	0.90	2	100,000	2
3	a<a	1	3 clientes	3	150,000	3
4	dd<dd	2	Variable	1	100,000	2
5	dd<dd	2	0.90	2	500,000	3
6	dd<dd	2	3 clientes	3	50,000	1
7	mixto	3	Variable	1	150,000	3
8	mixto	3	0.90	2	50,000	1
9	mixto	3	3 clientes	3	100,000	2

7. **Realizar el análisis estadístico e interpretar los resultados**. La normalidad de los datos fue evaluada con el estadístico Anderson - Darling [79] , como se muestra en la Figura 4.9, indicando que los datos siguen una distribución normal, ya que el indicador de valor p es de 0.451, siendo mayor que el nivel de significancia de 0.05.

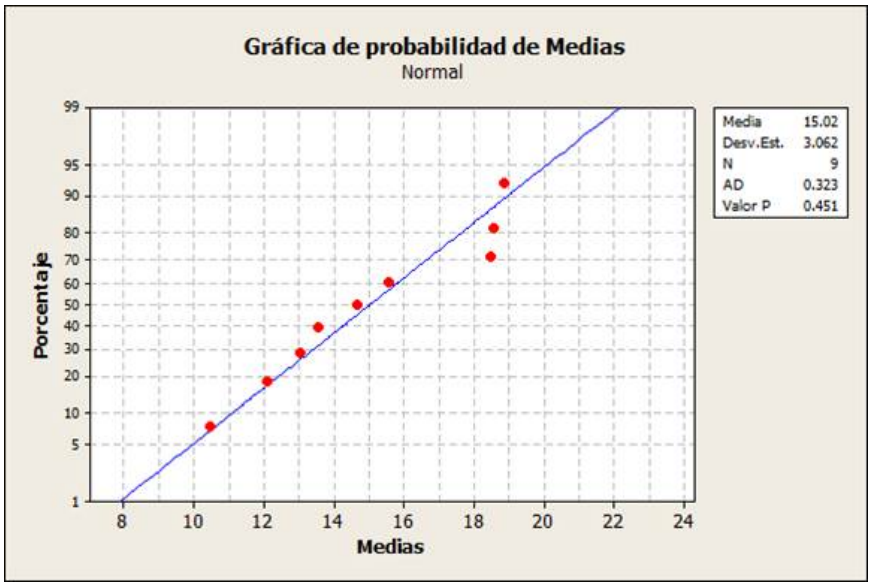


Figura 4.9: Evaluación de la normalidad de datos

Al hacer el análisis del diseño de Taguchi, los datos obtenidos de medias y señal a ruido, se muestran en la Tabla 4.9, utilizados posteriormente para generar las Figuras 4.10 y 4.11. La combinación más robusta de los niveles de los factores controlables es la que maximiza la razón señal a ruido S/R. Los valores de S/R se calcularon usando la fórmula lo más bajo es lo mejor, tal como se menciona

Tabla 4.8: Promedios del % Gap en cada una de las instancias

Instancias de prueba de 25 clientes						
Config.	***C1***	***C2***	***R1***	***R2***	***RC1***	***RC2***
1	3.78	17.59	0.03	10.26	0.44	8.03
2	0.68	2.17	**-0.63**	3.25	**-0.56**	0.22
3	3.53	53.52	0.008	8.25	0.22	11.63
4	0.90	58.90	0.89	3.07	3.30	5.25
5	0.55	36.97	**-1.44**	1.84	**-3.46**	0.72
6	1.49	13.80	1.27	1.87	3.30	7.68
7	1.27	71.01	0.58	2.82	3.30	7.89
8	0.90	0.62	**-0.25**	2.66	**-1.42**	0.35
9	1.17	57.36	**-0.92**	1.85	2.34	5.75
Instancias de prueba de 50 clientes						
1	16.14	23.67	7.97	25.85	2.47	26.72
2	13.42	6.26	8.52	19.96	3.31	16.5
3	13.57	10.20	5.88	22.09	1.71	24.33
4	7.34	14.25	11.72	17.86	6.55	17.79
5	10.72	6.41	7.46	18.42	1.70	15.9
6	8.33	31.97	10.90	21.07	6.70	17.6
7	6.13	8.93	11.06	18.48	7.03	20.20
8	16.80	13.96	8.70	19.62	4.20	14.50
9	5.63	24.85	9.85	20.63	6.84	19.93
Instancias de prueba de 100 clientes						
1	25.16	85.27		35.05		13.40
2	41.95	62.98		18.83		19.81
3	26.82	66.16		27.38		19.85
4	29.21	69.72		27.46		22.48
5	45.41	62.63		22.97		21.99
6	27.58	3.12		23.77		12.77
7	32.46	3.64		21.47		17.83
8	41.56	5.32		20.27		19.27
9	17.84	2.36		18.88		14.43

anteriormente, ya que se desea calcular la longitud total de la ruta más corta para el *G-VRPTW*.

Tabla 4.9: Resultados de medias y señal a ruido (S/R)

Configuración	***Medias***	***S/R***
1	18.86	-26.29
2	13.55	-24.80
3	18.45	-25.46
4	18.54	-26.06
5	15.55	-25.49
6	12.08	-19.83
7	14.63	-24.85
8	10.44	-21.18
9	13.05	-23.1727

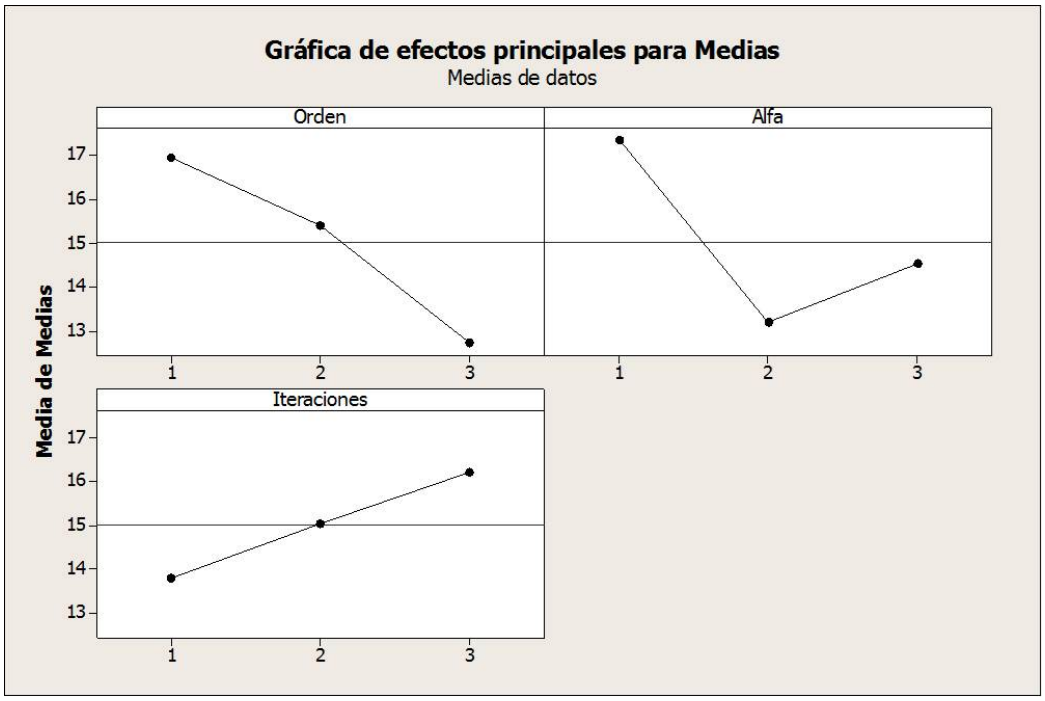


Figura 4.10: Gráfica de efecto principal para medias

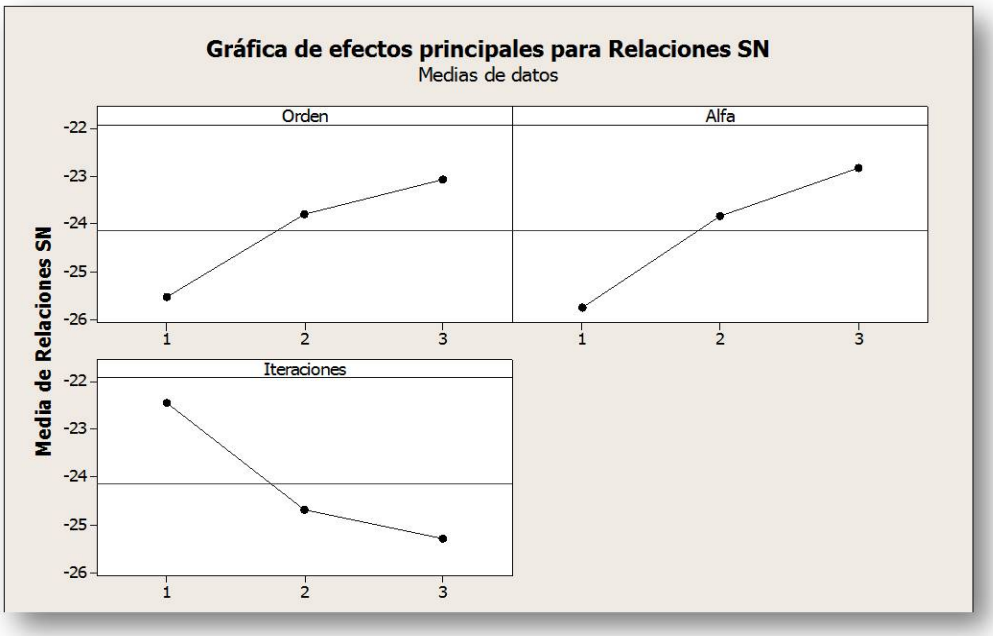


Figura 4.11: Gráfica de efecto principal para S/R

Por lo anterior la combinación en los factores que más afectan la S/R como se observa en Figura 4.11 son ordenamiento nivel 3 (mixto), alfa nivel 3 (3 clientes) e iteraciones nivel 1 (50,000), es decir, son los que más influyen en la variación de la variable de respuesta. De igual manera en la Figura 4.10 , se muestra que el factor ordenamiento en su nivel 3 (mixto), alfa en su nivel 2 (0.9) e iteraciones en el nivel 1 (50,000), son los que tienen más afecto sobre la media en el % Gap. Por lo tanto, la metodología Taguchi muestra que para obtener los % Gap con menores valores se recomienda establecer el nivel de cada parámetro como se define en la Tabla 4.10.

Tabla 4.10: Niveles sugeridos por Taguchi

Factor	Nivel
Ordenamiento	mixto
Alfa	0.90
Iteraciones	50000

8. **Realizar una prueba para confirmar el resultado**. En el conjunto que contiene el 20 % de las instancias y cuyo resultados se muestran en la Figura 4.12.A, se observó lo siguiente:

- En las instancias de 25 se reduce el Gap en 6.29 unidades y en 50 clientes se minimiza la distancia, ya que el % Gap obtenido disminuyó en 4.24 unidades al reportado en la Fase 1.
- En el comportamiento de las instancias de 100 clientes, el resultado muestra una disminución en 18.32 unidades, al compararlo con los valores obtenidos con calibración manual.

Al ejecutar el 80 % de las instancias, en la Figura 4.12.B, se observan los siguientes resultados:

- En los resultados de las instancias de 25 se reduce 2.88 unidades y en 50 clientes se logra minimizar la distancia, ya que el % Gap obtenido disminuyó en 1.03 unidades que el reportado en la **Fase 1**.
- En el comportamiento de las instancias de 100 clientes, el resultado muestra un incremento del 13.48 unidades, al compararlo con los valores obtenidos con calibración manual.

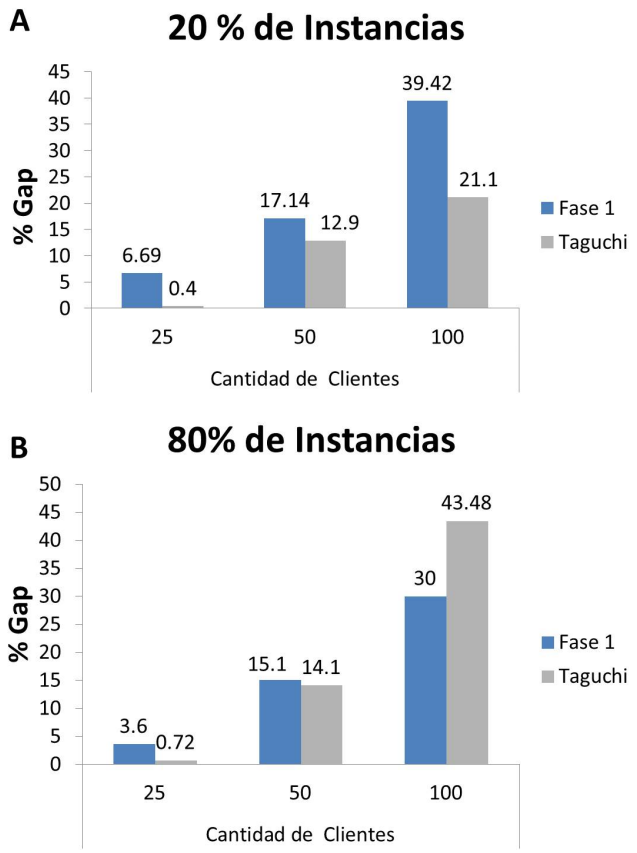


Figura 4.12: Gráfica comparativa de los resultados obtenidos. A en el 20 % de instancias y B en el 80 % de ellas

Al analizar los resultados obtenidos al ejecutar el algoritmo *G-VRPTW* con el vector de parámetros propuestos por Taguchi, mostrados en el Tabla 4.11, en donde el % Gap obtenido es producto de un promedio de las 10 replicas realizadas por cada una de las instancias, se puede apreciar que, en las instancias de 25 clientes, se ha logrado un Gap menor al 10 % por Taguchi, destacando que en dos grupos de instancias se ha mejorado la solución reportada en la literatura; tal es el caso de las instancias RC1 y RC2, que de acuerdo a las características de las instancias el algoritmo *G-VRPTW* favorece aquellas que su distribución georgráfica es mixta (RC), superando en todos los casos a los resultados obtenidos por la calibración empírica.

Los resultados en las instancias de 50 clientes reflejan valores variables, en un ran-

Tabla 4.11: Promedios del % Gap en el 80 % de las instancias

Resultados con Fase 1						
Config.	***C1***	***C2***	***R1***	***R2***	***RC1***	***RC2***
25	1.40	10.00	2.5	6.10	0.10	6.10
50	**8.60**	23.10	**7.20**	27.90	**7.60**	27.90
100	**25.80**	**33.10**	**13.30**	36.90	31.90	**36.90**
Resultados utilizando método Taguchi						
Config.	***C1***	***C2***	***R1***	***R2***	***RC1***	***RC2***
25	**0.25**	**0.97**	**1.67**	**2.77**	**-0.65**	**-0.68**
50	10.45	**15.23**	10.70	**17.72**	9.35	**21.15**
100	74.22	71.08	20.69	**35.10**	**24.20**	37.74

go entre 9.35 a 21.15 de % Gap, favoreciendo las instancias con ventanas estrechas y de horizonte de programación corto con el vector de parámetros de Taguchi; en tanto los resultados obtenidos por calibración manual, en tres casos lo superan en promedio por 1.03 unidades. En los resultados reflejados en las instancias de 100 clientes se observan que la mayoría están alejados de la mejor solución conocida en la literatura revisada. Para el grupo de instancias C2 los mejores resultados se presentan en las configuración 9 para esta clase de instancias, consultar Tabla 4.8 y la configuración 8 sugerida por Taguchi presentó, en el peor caso un Gap de 74.22 %, frente al 25.80 % obtenido por la configuración manual, lo que muestra un efecto negativo al considerar las medias, en un grupo de datos que no se ajusta totalmente a la normalidad Figura 4.9.

4.3.2. Resultados finales

Después de ejecutar *G-VRPTW* con los niveles de los factores sugeridos al emplear el método Taguchi, (en el 80 % de las instancias), se realizó un análisis de las iteraciones necesarias para obtener la mejor solución, y al ejecutar el algoritmo en el rango de 25,000 a 50,000 iteraciones, se observó que se generaron

el 51 % de las mejores soluciones. Derivado de ello, se ejecutó el algoritmo con los parámetros de orden y α ya establecidos por el método Taguchi, y se aumentó el número de iteraciones con el fin de explorar por una mejora en los resultados.

De éste ejercicio los resultados obtenidos al ejecutar el algoritmo con 150,000 iteraciones (150k) se reflejan en la Tabla 4.12, donde se aprecia que al aumentar el número de iteraciones se mejoran tanto al ejecutar al grupo del 20 % de las instancias como al grupo con el 80 % de las instancias.

Tabla 4.12: Promedio de % Gap por tipo de calibración

Instancia	Clientes	Fase 1	Taguchi	150k
20 %	25	6.69	0.40	-0.17
	50	17.14	12.90	11.02
	100	39.42	21.10	20.47
80 %	25	3.60	0.72	0.05
	50	15.13	14.10	10.68
	100	30.00	43.48	41.30

Capítulo 5

Discusiones y Conclusiones

En este capítulo se presentan las conclusiones basadas en los resultados obtenidos, las observaciones y experiencias. Se plantea el trabajo futuro derivado de esta investigación que contribuya a la solución de problemas de rutas de vehículos.

5.1. Discusiones

Una vez concentrando los resultados finales del *G-VRPTW*, estableciendo como parámetros un ordenamiento mixto, con α igual a 0.90 y 150,000 iteraciones, se hizo una comparación con los resultados que se han revisado en la literatura con otras investigaciones en las instancias de 100 clientes, Observando que el algoritmo *G-VRPTW* ha superado los resultados algunos de estos trabajos, sin embargo están alejados de los reportados como mejores en la literatura revisada.

Es importante aclarar que estos resultados del *G-VRPTW* son sin minimizar el número de vehículos, solo se consideró como función objetivo minimizar la distancia total de las rutas. En base a lo anterior el minimizar los vehículos es un objetivo que deberá considerarse como una mejora al algoritmo, y así poder ser comparado con igualdad de especificaciones con el resto de las investigaciones y las reportadas en la literatura como mejores soluciones.

Los trabajos de [60], [61] y de [59], a los que hace referencia la Tabla 5.1, son descritos en la Tabla 2.3. El trabajo de [36] construye rutas en paralelo, hace uso de la estructura de inserción genérica de Solomon, se inicializa un conjunto de rutas al mismo tiempo, y los clientes restantes se insertan en cualquiera de esas rutas. El trabajo de [80] resuelve el *VRPTW* utilizando la metaheurística de búsqueda Tabú.

Los resultados óptimos son mostrados por el tipo de instancia por diferentes trabajos, consultar [71]. Los resultados mostrados representan una aproximación a los paráme-

Tabla 5.1: Comparación con otras investigaciones en instancias de 100 clientes

Grupo inst.	Óp.	*GRASP* [60]	S-1987 [56]	Paral. [36]	Tabú [80]	*GRASP* [61]	GA [59]	G-VRPTW
R1	1210.3	1325.4	1436.7	1509.0	1294.5	1603.0	1192.1	**1512.6**
vehíc.	12.9	12.6	13.6	13.3	12.8	18.5	13.8	**17.7**
R2	951.03	1164.2	1402.4	1386.6	1163.5	1268.5	821.3	**1330.7**
vehíc.	2.7	3.1	3.3	3.1	3.2	4	4.9	6
C1	827.1	827.3	951.9	1343.6	870.9	885.0	833.3	**1278.4**
vehíc.	10	10	10	10.7	10	10.5	10	14
C2	587.9	589.6	692.7	797.5	611.0	598.0	666.2	972.5
vehíc.	3	3	3.1	3.4	3	3	3.4	5.8
RC1	1396.4	1500.9	1596.5	1723.7	1458.7	1721.0	1354.6	**1712.8**
vehíc.	11.5	12.6	13.5	13.4	12.8	15	10.8	17
RC2	1129.2	1414.2	1682.1	1651.0	1448.6		1014.6	**1546.0**
vehíc.	3.3	3.5	3.9	3.6	3.5		6	7.7

tros que se deben de considerar en el *G-VRPTW*, como se indica en [81] y se corroboró con la exploración adicional al modificar la cantidad de iteraciones. Aún con la anterior limitante se pudo observar que, el algoritmo presenta un alto rendimiento en las instancias de 25 clientes, como observamos en la Tabla 4.12, donde se indica que se logra mejorar el promedio de la mejor solución en la literatura revisada en cuanto a minimizar distancia, ya que no se ha considerado minimizar el número de vehículos. En las instancias de 50 clientes, en promedio se tiene un Gap del 10 % respecto a la mejor solución y en las instancias de 100 clientes, es indiscutible la necesidad de explorar la superficie en base a los resultados obtenidos en el presente trabajo para reducir el diferencial existente con la mejor solución conocida.

Finalmente como se aprecia en la Figura 5.1, los resultados obtenidos muestran que al establecer el vector de parámetros mediante un método de calibración mejora el desempeño del algoritmo *G-VRPTW*, tal como se aprecia al comparar la **Fase 1**, con un Gap de 18.7 %, con el obtenido al utilizar el método Taguchi descendiendo al 16.6 %. Al aumentar el número de iteraciones y mantener el vector de parámetros Taguchi, el Gap disminuye aún más acercándo los resultados finales un 13.9 % en promedio de la mejor solución revisada en la literatura.

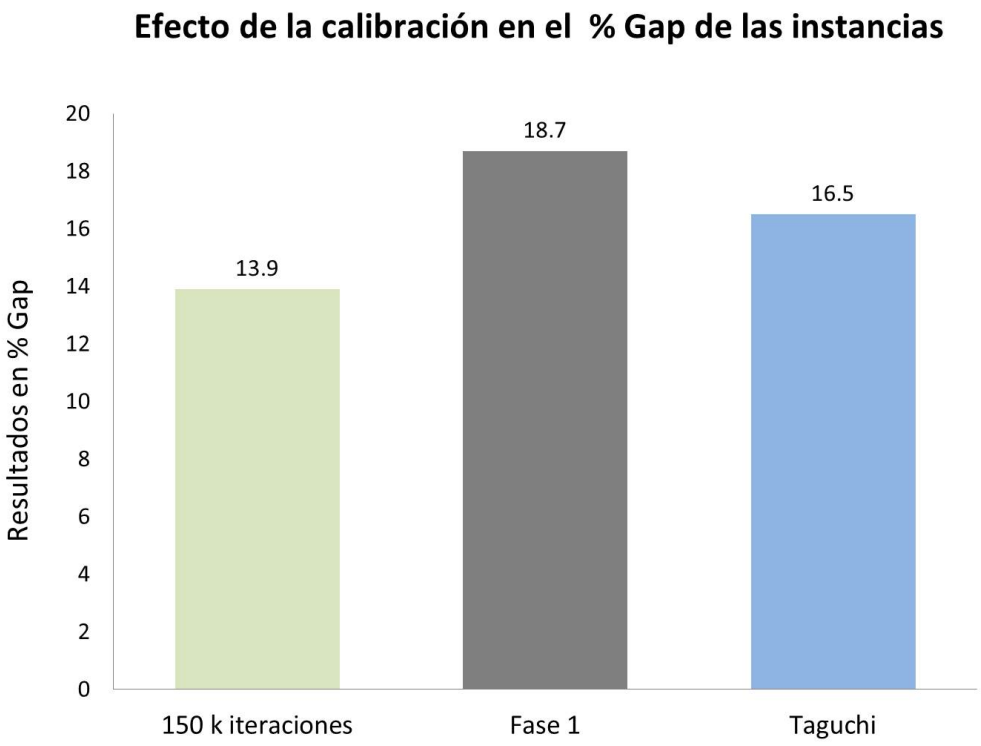


Figura 5.1: Gráfica comparativa de los resultados finales obtenidos en las aproximaciones a la resolución del problema

5.2. Conclusiones

Realizando una revisión de los objetivos específicos que se plantearon al inicio de ésta investigación, se presenta en la Sección 2, el estudio de las investigaciones que se encontraron en la literatura relacionadas con el problema de rutas de vehículos con ventanas abiertas, y aquellos en los que se utilizó la metaheurística *GRASP*, cumpliendo de esta forma con el objetivo de realizar un estudio de los diferentes problemas, moldeamientos y técnicas de solución empleadas para resolver el problema de rutas de vehículos, en particular con la variante de ventanas de tiempo que utilizara la metaheurística *GRASP*.

De igual forma al realizar la programación y ejecución del algoritmo en C++ para dar soluciones al *VRPTW*, se realizó una comparación con los mejores resultados que se han registrado en la literatura [71] y con las diferentes investigaciones encontradas que utilizaron *GRASP*, mostrados en la Tabla 5.1, aclarando que al *G-VRPTW* es necesario incluirle la función objetivo de minimizar vehículos para ser comparados con ellos en igualdad de circunstancias.

Al cumplir con el objetivo general de aplicar un algoritmo metaheurístico para problemas de rutas de vehículos con ventanas de tiempo, estableciendo como función objetivo minimizar distancias, se utilizó *GRASP*, generando el algoritmo que denominamos *G-VRPTW*, se aplicó el método Taguchi para calibrar los parámetros de entrada en la fase de construcción, cumpliendo así en su totalidad con los objetivos planteados en este trabajo.

Las recomendaciones que se concluyen de ésta investigación es que al utilizar la metaheurística *G-VRPTW*, el vector de parámetro debe contener un α con 0.90, ya que de acuerdo a los resultados obtenidos este parámetro nos permite mejorar los resultados al compararlo con la selección de 3 clientes que se recomienda en la literatura, para problemas similares.

Una vez utilizado el método Taguchi, se recomienda efectuar un análisis de superficie de respuesta con el fin de establecer los parámetros que mejoren los resultados del algorítmo *G-VRPTW*, sobre todo para las instancias de 100 clientes, como se realiza en la propuesta de la *herramienta de Calibra*, que utiliza primero la metodología de Taguchi y después aplica una búsqueda local [81]. Es recomendable incluir el objetivo de minimizar vehículos con el fin de buscar mejores soluciones.

5.3. Trabajo futuro

Como trabajos futuros se propone incluirle al algoritmo *G-VRPTW* una metaheurística que combinada con el *GRASP* mejore los resultados obtenidos. Se sugiere, incluir otros indicadores de desempeño diferentes a la calidad de la solución, para ser competitivo con el resto de los trabajos conocidos en la literatura como por ejemplo minimizar el número de vehículos, considerándolo como una función objetivo adicional, considerar el esfuerzo computacional y la robustez de los parámetros.

Además se sugiere asociarle una metaheurística que mejore el rendimiento del *G-VRPTW*, es decir considerar estas soluciones que se obtuvieron como datos iniciales de un segundo algoritmo.

El llevar el *G-VRPTW* a casos prácticos, sería de igual forma un trabajo futuro a considerar. En donde los datos iniciales serían los de alguna empresa que requiera satisfacer a sus clientes en un horario establecido y aplicarlo así a un software comercial para las pequeñas y medianas empresas.

Por último considerando las nuevas tendencias del cuidado del medio ambiente se puede plantear solucionar el problema del *VRPTW* considerando vehículos eléctricos.

5.4. Aportaciones académicas

1. Artículo aceptado: *Efecto de la calibración de parámetros mediante un diseño Taguchi en el algoritmo GRASP resolviendo el problema de rutas de vehículos con restricciones de tiempo*, Revista Computación y Sistemas ISSN-2007-9737, 2017. Alma Danisa Romero Ocaño, María de los Ángeles Cosío León, Víctor Manuel Valenzuela Alcaraz, Gener Avilés Rodríguez, Anabel Martínez Vargas.

2. Artículo publicado: *Problemas de rutas abiertas de vehículos con restricciones de capacidad heterogéneas*, Revista I+D (Innovación más Desarrollo), Vol. 12, Instituto Tecnológico de Nogales, Diciembre 2015. Héctor Efraín Ruiz y Ruiz, Alma Danisa Romero Ocaño, Víctor Manuel Valenzuela Alcaraz.

3. Conferencia: *Diseño de software logístico para PYMES*, segundo congreso INNOVA Empresarial 20-15, Noviembre 2015, Agua Prieta, Sonora.

Bibliografía

[1] David M Martin Bovet and W Bob. Camino sin obstáculos. *GestiónVol. 5, no. 1 (2000 ene.), p. 70-74*, 2000.

[2] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[3] WW Garvin, HW Crandall, JB John, and RA Spellman. Applications of linear programming in the oil industry. *Management Science*, 3(4):407–430, 1957.

[4] Michel L Balinski and Richard E Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

[5] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[6] Merrill M Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[7] HGM Pullen and MHJ Webb. A computer application to a transport scheduling problem. *The Computer Journal*, 10(1):10–13, 1967.

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

[9] Linda Bibiana Rocha Medina, Elsa Cristina González La Rotta, and Javier Arturo Orjuela Castro. Una revisión al estado del arte del problema de ruteo de vehículos: Evolución histórica y métodos de solución. *Ingeniería*, 16(2):35–55, 2011.

[10] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[11] Miguel Cárdenas-Montes. Predicting hardness of travelling salesman problem instances. In *Conference of the Spanish Association for Artificial Intelligence*, pages 68–78. Springer, 2016.

[12] Rubén Iván Bolaños, Eliana MO Toro, and Mauricio E Granada. A population-based algorithm for the multi travelling salesman problem. *International Journal of Industrial Engineering Computations*, 7(2):245, 2016.

[13] O Toro, M Eliana, Z Escobar, H Antonio, E Granada, et al. Literature review on the vehicle routing problem in the green transportation context. *Luna Azul*, (42):362–387, 2016.

[14] Yannis Marinakis and Magdalene Marinaki. A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. *Computers & Operations Research*, 37(3):432–442, 2010.

[15] Beatrice Ombuki, Brian J Ross, and Franklin Hanshar. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24(1):17–30, 2006.

[16] Brian Kallehauge, Jesper Larsen, Oli BG Madsen, and Marius M Solomon. Vehicle routing problem with time windows. In *Column generation*, pages 67–98. Springer, 2005.

[17] Qie He and Yongjia Song. Vehicle routing problems with time windows and convex node costs. 2016.

[18] Ali Kourank Beheshti and Seyed Reza Hejazi. A novel hybrid column generation-metaheuristic approach for the vehicle routing problem with general soft time window. *Information Sciences*, 316:598–615, 2015.

[19] Pedro Pablo Ballesteros Silva and Antonio Hernando Escobar Zuluaga. Description of the classification of publications and the models used in solving of the vehicle routing problem with pickup and delivery. *Revista de Ingenierías: Universidad de Medellín*, 15(28):14, 2016.

[20] Ali Asghar Rahmani Hosseinabadi, Javad Vahidi, Valentina Emilia Balas, and Seyed Saeid Mirkamali. Ovrp_gels: solving open vehicle routing problem using the gravitational emulation local search algorithm. *Neural Computing and Applications*, pages 1–14.

[21] Panagiotis P Repoussis, Christos D Tarantilis, and George Ioannou. The open vehicle routing problem with time windows. *Journal of the Operational Research Society*, 58(3):355–367, 2007.

[22] Julio Brito, F Javier Martínez, José Andrés Moreno, and José L Verdegay. An aco hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints. *Applied Soft Computing*, 32:154–163, 2015.

[23] Diego Cattaruzza, Nabil Absi, Dominique Feillet, and Thibaut Vidal. A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236(3):833–848, 2014.

[24] Jairo R Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, and Nilson Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129, 2015.

[25] Jorge Oyola, Halvard Arntzen, and David L Woodruff. The stochastic vehicle routing problem, a literature review, part ii: solution methods. *EURO Journal on Transportation and Logistics*, pages 1–40, 2016.

[26] Víctor Yepes Piqueras. *Optimización heurística económica aplicada a las redes de transporte del tipo VRPTW.* PhD thesis, 2008.

[27] KW Knight and JP Hofer. Vehicle scheduling with timed and connected calls: A case study. *Journal of the Operational Research Society*, 19(3):299–310, 1968.

[28] Oli BG Madsen. Variable splitting and vehicle routing problems with time windows. *Preprint IB/1988. IMSOR, The Technical University of Denmark, Lyngby, Denmark*, 1988.

[29] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[30] Niklas Kohl. Exact methods for time constrained routing and related scheduling problems. 1995.

[31] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. 1999.

[32] Tan Kay Chen, Lee Loo Hay, and Ou Ke. Hybrid genetic algorithms in solving vehicle routing problems with time window constraints. *Asia-Pacific Journal of Operational Research*, 18(1):121, 2001.

[33] Jörg Homberger and Hermann Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 37(3):297–318, 1999.

[34] Wen-Chyuan Chiang and Robert A Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27, 1996.

[35] Yves Rochat and Éric D Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of heuristics*, 1(1):147–167, 1995.

[36] Jean-Yves Potvin and Jean-Marc Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

[37] Robert A Russell. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation science*, 29(2):156–166, 1995.

[38] Sam R Thangiah. A hybrid genetic algorithms, simulated annealing and tabu search heuristic for vehicle routing problems with time windows. *Practical handbook of genetic algorithms*, 3:347–381, 1999.

[39] Kay Chen Tan, Loo Hay Lee, QL Zhu, and Ke Ou. Heuristic methods for vehicle routing problem with time windows. *Artificial intelligence in Engineering*, 15(3):281–295, 2001.

[40] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

[41] Byung-In Kim, Seongbae Kim, and Surya Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.

[42] Alberto V Donati, Roberto Montemanni, Norman Casagrande, Andrea E Rizzoli, and Luca M Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European journal of operational research*, 185(3):1174–1191, 2008.

[43] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[44] Reza Eshtehadi, Mohammad Fathian, and Emrah Demir. Robust solutions to the pollution-routing problem with demand and travel time uncertainty. *Transportation Research Part D: Transport and Environment*, 51:351–363, 2017.

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

[46] Alfredo Olivera. Heurísticas para problemas de ruteo de vehículos. *Reportes Técnicos 04-08*, 2004.

[47] JAQUE PIRABÁN. Ra métodos aproximados para la solución del problema de enrutamiento de vehículos, 2008.

[48] Jin Qin, Yong Ye, Bi-rong Cheng, Xiaobo Zhao, and Linling Ni. The emergency vehicle routing problem with uncertain demand under sustainability environments. *Sustainability*, 9(2):288, 2017.

[49] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[50] Pierre A Devijver and Josef Kittler. *Pattern recognition theory and applications*, volume 30. Springer Science & Business Media, 2012.

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

[52] Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[53] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

[54] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–72, 1992.

[55] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[56] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[57] Andrew Lim and Fan Wang. A smoothed dynamic tabu search embedded grasp for m-vrptw. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 704–708. IEEE, 2004.

[58] Zhiye Li, Songshan Guo, Fan Wang, and Andrew Lim. Improved grasp with tabu search for vehicle routing with both time window and limited number of vehicles. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 552–561. Springer, 2004.

[59] Yong Mao and Yanfang Deng. Solving vehicle routing problem with time windows with hybrid evolutionary algorithm. In *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, volume 1, pages 335–339. IEEE, 2010.

[60] George Kontoravdis and Jonathan F Bard. A grasp for the vehicle routing problem with time windows. *ORSA journal on Computing*, 7(1):10–23, 1995.

[61] Wanpracha Chaovalitwongse, Dukwon Kim, and Panos M Pardalos. Grasp with a new local search scheme for vehicle routing problems with time windows. *Journal of Combinatorial Optimization*, 7(2):179–207, 2003.

[62] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[63] Mansour Alssager and Zulaiha Ali Othman. Taguchi-based parameter setting of cuckoo search algorithm for capacitated vehicle routing problem. In *Advances in Machine Learning and Signal Processing*, pages 71–79. Springer, 2016.

[64] Panagiotis P Repoussis, Dimitris C Paraskevopoulos, Christos D Tarantilis, and George Ioannou. A reactive greedy randomized variable neighborhood tabu search for the vehicle routing problem with time windows. In *International Workshop on Hybrid Metaheuristics*, pages 124–138. Springer, 2006.

[65] Julio Brito, F Javier Martínez, José Andrés Moreno, and José L Verdegay. A grasp–vns hybrid for the fuzzy vehicle routing problem with time windows. In *International Conference on Computer Aided Systems Theory*, pages 825–832. Springer, 2009.

[66] Syrine Roufaida Ait Haddadene, Nacima Labadie, and Caroline Prodhon. A grasp× ils for the vehicle routing problem with time windows, synchronization and precedence constraints. *Expert Systems with Applications*, 66:274–294, 2016.

[67] Jesica de Armas, Eduardo Lalla-Ruiz, Christopher Expósito-Izquierdo, Dario Landa-Silva, and Belén Melián-Batista. A hybrid grasp-vns for ship routing and scheduling problem with discretized time windows. *Engineering Applications of Artificial Intelligence*, 45:350–360, 2015.

[68] Gerald Senarclens de Grancy and Marc Reimann. Vehicle routing problems with time windows and multiple service workers: a systematic comparison between aco and grasp. *Central European Journal of Operations Research*, 24(1):29–48, 2016.

[69] Masoud Yaghini and Mohammad Rahim Akhavan Kazemzadeh. Dimma: A design and implementation. *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends: Advancements and Trends*, page 90, 2012.

[70] M Rajkumar, P Asokan, N Anilkumar, and T Page. A grasp algorithm for flexible job-shop scheduling problem with limited resource constraints. *International Journal of Production Research*, 49(8):2409–2423, 2011.

[71] Marius M Solomon. Vrptw benchmark problem. http://web.cba.neu.edu/ msolomon/problems.htm, 2005.

[72] Steven P. Coy, Bruce L. Golden, George C. Runger, and Edward A. Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, January 2001.

[73] Thomas Bartz-Beielstein. How experimental algorithmics can benefit from mayo's extensions to neyman–pearson theory of testing. *Synthese*, 163(3):385–396, 2008.

[74] Fu Gu, Philip Hall, Nicholas J Miles, Qianwen Ding, and Tao Wu. Improvement of mechanical properties of recycled plastic blends via optimizing processing parameters using the taguchi method and principal component analysis. *Materials & Design (1980-2015)*, 62:189–198, 2014.

[75] John Morgan, Jiju Antony, Daniel Perry, Chengbo Wang, and Maneesh Kumar. An application of taguchi method of experimental design for new product design and development process. *Assembly Automation*, 26(1):18–24, 2006.

[76] Genichi Taguchi, Subir Chowdhury, and Yuin Wu. *Taguchi's quality engineering handbook*. Wiley, 2005.

[77] Jorge Limon-Romero, Diego Tlapa, Yolanda Baez-Lopez, Aide Maldonado-Macias, and Leonardo Rivera-Cadavid. Application of the taguchi method to improve a medical device cutting process. *The International Journal of Advanced Manufacturing Technology*, 87(9-12):3569–3577, 2016.

[78] Humberto Gutiérrez Pulido, Román De la Vara Salazar, Porfirio Gutiérrez González, Carlos Téllez Martínez, and María del Carmen Temblador Pérez. *Análisis y diseño de experimentos*. McGraw-Hill, 2004.

[79] Theodore W Anderson and Donald A Darling. Asymptotic theory of certain"goodness of fitçriteria based on stochastic processes. *The annals of mathematical statistics*, pages 193–212, 1952.

[80] Jean-Yves Potvin, Tanguy Kervahut, Bruno-Laurent Garcia, and Jean-Marc Rousseau. The vehicle routing problem with time windows part i: tabu search. *INFORMS Journal on Computing*, 8(2):158–164, 1996.

[81] Belarmino Adenso-Diaz and Manuel Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations research*, 54(1):99–114, 2006.