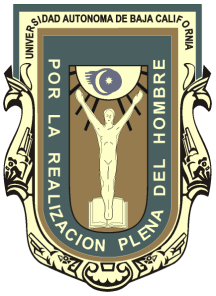
Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño



Maestría y Doctorado en Ciencias e Ingeniería

Algoritmo metaheurístico GRASP aplicado al problema de ruteo por arcos con capacidad limitada en los vehículos

TESIS

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

MAESTRO EN INGENIERÍA

Presenta

VÍCTOR MANUEL VALENZUELA ALCARAZ

Ensenada, Baja California, Julio del 2017.

Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Algoritmo metaheurístico GRASP aplicado al problema de ruteo por arcos con capacidad limitada en los vehículos

TESIS

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

MAESTRO EN INGENIERÍA

Presenta

Víctor Manuel Valenzuela Alcaraz

Aprobada por:

Dra. María de los Ángeles Cosío León
Directora de Tesis

Dra. Anabel Martínez Vargas
Miembro del Comité

Dr. Alvaro Alberto López Lambraño
Miembro del Comité

Dr. Miguel Enrique Martínez Rosas
Miembro del Comité

Ensenada, Baja California, Julio del 2017.

Resumen de la tesis de **Víctor Manuel Valenzuela Alcaraz**, presentada como requisito parcial para la obtención del grado de MAESTRO EN INGENIERÍA del programa de Maestría y Doctorado en Ciencias e Ingeniería (MYDCI) de la UABC. Ensenada Baja California, México, Mayo del 2017.

Algoritmo metaheurístico GRASP aplicado al problema de ruteo por arcos con capacidad limitada en los vehículos

Resumen Aprobado por:

Dra. María de los Ángeles Cosio León
Tutora

En esta tesis, la metaheurística GRASP (Greedy Randomized Adaptive Search Procedure) es aplicada al problema de ruteo en arcos con capacidad limitada en los vehículos CARP (Capacitated Arc Routing Problem), el cual consiste en atender las demandas en un conjunto de calles en una localidad con un conjunto de vehículos con capacidad limitada en su carga, iniciando y terminando en un mismo depósito. El objetivo del problema es reducir el costo (distancia o tiempo) del recorrido de tal forma que las demandas correspondientes al conjunto de calles sea atendida por un solo vehículo y que cada uno de los vehículos no excedan su capacidad. Los resultados son comparados con instancias conocidas en la literatura.

Palabras Clave: *CARP*, *Problema de Ruteo en Arcos con Capacidad Limitada en los Vehículos*, *GRASP*, *metaheurística.*

Dedicatoria

A Dios por haberme guiado por el camino de la felicidad, brindarme salud, bienestar y una excelente familia.

A mi esposa, por brindarme su apoyo y motivación día con día para alcanzar nuevas metas, tanto profesionales como personales, pero sobre todo por el amor y cariño que siempre me ha manifestado durante estos 19 años, gracias por estar a mi lado y ser parte de mi vida.

A mi dos hijas, quienes me prestaron el tiempo que les pertenecía para cumplir con esta meta, siempre las cuidaré para verlas hechas personas capaces y que puedan valerse por si mismas.

A mis padres quienes me dieron vida, apoyo, educación y consejos, a ustedes por siempre mi corazón y mi agradecimiento.

A mis hermanas, cuñadas, cuñados y sobrinos por estar conmigo y ser parte de mi vida, los quiero mucho.

Agradecimientos

Quiero expresar mi agradecimiento:

A mi directora de tesis **Dra. María de los Ángeles Cosío León**, a mis sinodales **Dra. Anabel Martínez Vargas, Dr. Miguel Enrique Martínez Rosas y Dr. Álvaro Alberto López Lambraño** por ofrecerme sus consejos, criticas, su paciencia, apoyo e interés en el desarrollo de esta tesis.

A mis compañeros y amigos por su amistad y los buenos momentos.

A mis maestros, gracias por su tiempo, por su apoyo, así como por el conocimiento que me transmitieron en el desarrollo de mi formación profesional.

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

Al Tecnológico Nacional de México, que a través del Instituto Tecnológico de Agua Prieta (ITAP), por otorgar la posibilidad de desarrollarme de manera profesional y académica.

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

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Constantemente, empresas públicas y privadas deben plantearse problemas de decisión relacionados con la planeación y diseño de rutas como la recolección de basura, reparto de correspondencia, limpieza o mantenimiento de calles, distribución de productos, transporte de personal, lectura de medidores de energía eléctrica, entre otros. En términos generales, todas estas situaciones son consideradas como problemas de ruteo y pueden ser formulados como problemas de optimización [1].

Dentro de los problemas de optimización se encuentran los de optimización combinatoria, donde sus soluciones se codifican usando variables discretas y el proceso de búsqueda consiste en explorar un espacio de soluciones (del problema) la cual puede ser representado mediante listas, conjuntos, matrices o grafos [2].

De igual manera los problemas de rutas pueden ser representados por grafos, donde los lugares a visitar se refiere a los vértices a los que se debe transportar bienes o servicios. Así mismo dos vértices forman un arco o arista y simbolizan las calles, carreteras, cables de transmisión, redes hidráhulicas, etc., que funcionan como vía de comunicación. En general, el objetivo de los problemas de rutas consisten en optimizar los costos expresados en reducción o minimización de distancia, tiempo, costo, etc., En este trabajo, el costo de una ruta se refiere a la distancia total recorrida por el vehículo.

De acuerdo a como se da servicio a la demanda, los problemas de rutas pueden ser clasificados en problemas de ruteo por vértices (*VRP o Vehicle Routing Problems*) y problemas de ruteo por arcos (*ARP o Arc Routing Problems*).

El *VRP* consiste en diseñar rutas para una flota de vehículos, que dan servicio a un conjunto de vértices (clientes, tiendas, escuelas, ciudades, almacenes, etc.) con el menor costo [3]. Mientras *ARP* es diseñar rutas para una flota de vehículos, que dan servicio a un conjunto de arcos (recolección de basura, entrega de correo, barrido de calles, rutas de autobús escolar, etc.) con el menor costo [4]. Ambos, *ARP* y *VRP* tienen muchos variantes dependiendo del tipo de problema que se desea resolver.

En el caso particular de este trabajo de investigación, se aplica un algoritmo metaheurístico que resuelve un tipo especial de *ARP* llamado problema de rutas en arcos con capacidad limitada en los vehículos (*CARP*) propuesto por Golden y Wong [5].

1.1. Planteamiento del problema

El problema que se aborda en este trabajo de investigación está inspirado en el problema de una empresa recolectora de residuos sólidos urbanos, la cual requiere optimizar sus rutas, ya que en los últimos años, sus costos de operación se han incrementado debido al aumento de: combustibles, kilómetros recorridos, número de vehículos, tiempo de los recorridos, etc. Problema que puede ser formulado y analizado como el problema de ruteo en arcos con capacidad limitada en los vehículos (*CARP*).

El *CARP* atiende las demandas sobre determinadas calles de una red vial a través de una flota homogénea de vehículos, los cuales inician y finalizan sus recorridos en un único depósito [6]. La función objetivo del problema es minimizar el costo total de recorrido de manera que se atiendan todas las demandas sin exceder la capacidad de carga de los vehículos utilizados.

Uno de los beneficios de estudiar y aplicar éste problema, es su aplicación real en diversos tipos de empresas que incluyan operaciones de transporte y/o distribución de bienes y servicios. No obstante, es probable que dichas empresas posean características adicionales al *CARP*; es decir, características relacionadas con su contexto que deben ser consideradas y por tanto, agregadas al modelo original y así reflejar mejor su realidad; por ejemplo, minimizar el número de vehículos usados, cumplir con las demandas dentro de un cierto intervalo de tiempo especificado, considerar una red vial con calles en un solo sentido, en dos o ambos, utilizar varios depósitos desde donde los vehículos puedan comenzar (y finalizar) sus recorridos, considerar vehículos con diferentes capacidades de carga, entre otras.

1.2. Justificación

La importancia de aplicar algoritmos metaheurísticos que resuelvan este tipo de problemas radica en su aplicación para organismos públicos y empresas privadas, en ambos casos la distribución de bienes y servicios desempeña un papel crucial y constituye una gran parte de sus gastos totales. Por ello, el estudio de estos problemas esta motivado por la necesidad de ofrecer mejores y variadas alternativas de solución a aquellas organizaciones que deben tratar cotidianamente con problemas de ruteo y logística [7]. Además, este problema pertenecen a la clase *NP-Difícil* [8], es decir, se requiere de un tiempo de computo elevado incluso para problemas relativamente pequeños y de poca utilidad practica. Es por esto que los algoritmos metaheurísticos han tenido gran

relevancia en la solución de este tipo de problemas, gracias a su facilidad de implementación, la calidad de los resultados obtenidos y el bajo tiempo computacional.

Como se detalla más adelante en la sección 2.3.3, dependiendo se sus características, las metaheurísticas se pueden clasificar según su origen, el número de soluciones utilizadas en el proceso de búsqueda, uso de la función objetivo, la cantidad de estructuras de vecindario, su uso o no de memoria y finalmente, según su forma en que se construya la solución o exploran el vecindario de una solución [9].

La metaheurística utilizada como alternativa de solución para este trabajo posee dos de las característica arriba mencionadas, la de trabajar soluciones únicas durante el proceso de búsqueda (método de trayectoria) y la forma aleatoria de construir la solución. Esta elección está motivada por el hecho de que estas características permiten una descripción más clara sobre el desempeño del algoritmo con respecto a la instancia de prueba además de proporcionar soluciones iniciales diversas y de buena calidad.

1.3. Objetivo general

Analizar la calidad de las soluciones que puede proveer el algoritmo metaheurístico GRASP, al problema de ruteo por arcos con capacidad limitada en los vehículos (CARP).

1.4. Objetivos específicos

- *Estudiar el problema de ruteo por arcos con capacidad limitada en los vehículos (CARP).*
- *Analizar el modelado y técnicas empleadas por la literatura, para proponer soluciones al problema de ruteo por arcos con capacidad limitada en los vehículos (CARP), utilizando la metaheurística GRASP.*
- *Analizar y ajustar el algoritmo GRASP, para buscar soluciones al problema de ruteo por arcos con capacidad limitada en los vehículos (CARP).*
- *Calibrar los parámetros del algoritmo GRASP considerando con instancias de prueba propuestas por M. Kiuchi (Kshs) [10], B.L. Golden (Gdb) [5] y E. Benavent (Val) [11].*
- *Comparar los resultados ofrecidos por el algoritmo GRASP, una vez calibrado, con las soluciones propuestas en la literatura a las instancias de prueba propuestas por M. Kiuchi (Kshs) [10], B.L. Golden (Gdb) [5] y E. Benavent (Val) [11].*

1.5. Secuencia de la tesis

La presente tesis está estructurada de la siguiente forma:

Capítulo 1 En este capítulo se da una introducción a los problemas de rutas, se muestra el planteamiento del problema, se justifica la investigación, se plantean el objetivo general, los objetivos específicos y se da una breve descripción de los capítulos de los que comprende este trabajo.

Capítulo 2 En el capítulo dos se presenta una revisión de literatura, iniciando con la definición de los problemas de ruteo por arcos, se describe el modelo matemático para el *CARP*, se describen los métodos de solución para el *CARP* y las variantes que tiene éste tipo de problemas. Al final, se describe la metaheurística GRASP para la solución de problemas de rutas.

Capítulo 3 En el capítulo tres se describe la metodología *DIMMA*, posteriormente se muestra el algoritmo propuesto para la solución del CARP mediante el algoritmo *GRASP*. Se explican los operadores de búsquedas locales que se propone en la etapa de mejora del *GRASP*. La metodología propuesta para la solución del problema se describe también en este capítulo.

Capítulo 4 En este capítulo se muestran los experimentos computaciones realizados para evaluar el algoritmo propuesto, utilizando diferentes instancias de la literatura, al final del capítulo, se presenta un análisis de los resultados obtenidos.

Capítulo 5 Finalmente, el capítulo quinto está dedicado a las discusiones y conclusiones de la investigación, para terminar, se plantean posibles trabajos futuros.

Capítulo 2

Revisión de literatura

En este capítulo se presenta una revisión de literatura, iniciando con la definición y el origen del problema de ruteo por arcos, se describe el modelo matemático para el *CARP*, se describen los métodos de solución para el *CARP* y las variantes que tiene este tipo de problemas. Al final, se describe la metaheurística *GRASP* para la solución de problemas de rutas.

2.1. Problema de ruteo por arcos

Como se muestra en la figura 2.1 el problema de ruteo por arcos (ARP) se puede definir mediante un grafo $G = [V, A]$, donde V es un conjunto finito de puntos llamados vértices o nodos y A un conjunto de pares de vértices no ordenados llamados aristas.
Se dice que un grafo G es *dirigido* u orientado cuando los elementos de A tienen una dirección. A estos elementos se les llaman arcos y su dirección es representada con una flecha.
Por el contrario un grafo G se dice que es *no dirigido* si los elementos de A no tienen dirección. Estos elementos se les llaman aristas.
Así mismo, un grafo G se dice que es *conexo* si existe un camino para cualquier par de vértices.

El estudio de los ARP inicio con la presentación del problema del puente de Kõnigsberg en el año de 1735 por Leonhard Euler [12] como se muestra en la Figura 2.2. La pregunta que se hacían los lugareños era: habrá algún recorrido que permita atravesar los siete puentes, pasando por cada uno de ellos sólo una vez y regresando al mismo punto de origen?. Esta pregunta fue resuelta por Euler en 1736 [12] y cuya resolución dio origen a la teoría de grafos.

Este problema consistía en encontrar un recorrido en un conjunto de siete puentes que comunicaba la ciudad con dos islas, iniciando en algún punto de la ciudad, pasando solamente una vez por cada puente y regresando al punto de partida.

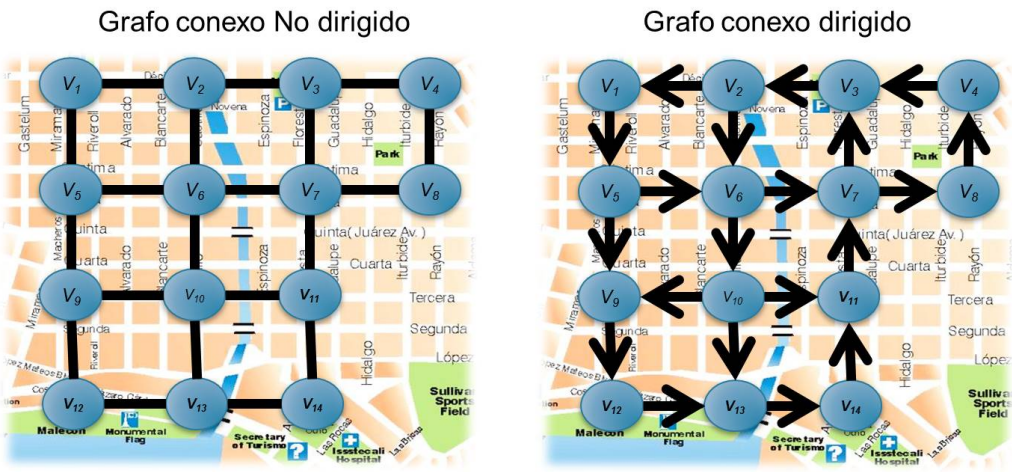


Figura 2.1: $G = [V, A]$ representa la red vial, $|V| = V_1, V_2, ..., V_{14}$ representan las esquinas y $|A| = (V_1, V_2), (V_2, V_3)...(V_{13}, V_{14})$ representan las calles. En el grafo no dirigido los elementos de $|A|$ no tienen dirección y se les llama aristas, por otro lado, en el grafo dirigido los elementos de $|A|$ tienen una dirección y se llaman arcos.

L. Euler demostró que es posible obtener este recorrido, siempre y cuando el número de calles incidentes en cada punto a visitar es par [12].
El Problema de los puentes de Kõnigsberg se puede plantear con un grafo conexo $G = [V, A]$, donde $|V|$ es el conjunto de vértices, $|A|$ el conjunto de aristas y consiste encontrar un recorrido que visite cada arista de $|A|$ exactamente una vez o bien informar que no es posible obtenerlo.

Figura 2.2: Los 7 puentes de Kõnigsberg.

En 1962 Kwan Mei-Ko [13] planteó el problema del cartero chino (*Chinese Postman Problem* o *CPP*), donde se debe de encontrar una ruta con la menor distancia posible de un cartero que debe repartir la correspondencia recorriendo al menos una vez cada una de las calles de una población, iniciando y terminando en la oficina de correos. Este problema se puede definir como un grafo conexo no dirigido $G = (V, A, C)$, donde C es

la matriz de costos, el problema consiste en encontrar un recorrido tal que visite cada arista al menos una vez al menor costo posible.

Posteriormente, surge el Problema del Cartero Rural (*Rural Postman Problem o RPP*) propuesto por Orloff en 1974 [14], este es una variante del *CPP*, con la diferencia de que el cartero debe repartir la correspondencia solo a un subconjunto de las calles (aristas requeridas). El *RPP* se define de la siguiente manera: sea una grafo conexo no dirigido $G = (V, A, C)$ donde C es una matriz de costos, encontrar un recorrido de costo mínimo tal que este visite cada una de las arista requeridas al menos una vez.
Como se indica en la Figura 2.3, la diferencia entre *CPP* y *RPP* es que en el problema del Cartero Rural solo un conjunto de aristas deben ser visitadas.

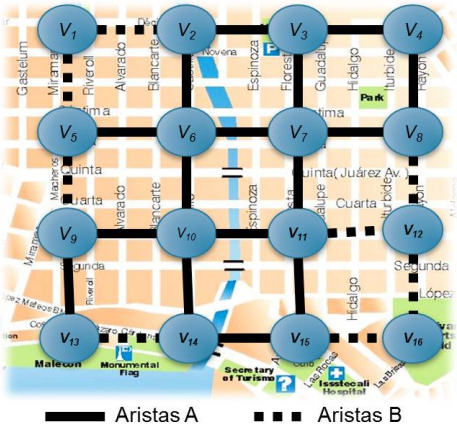


Figura 2.3: El *CPP* debe recorrer al menos una vez tanto las aristas A como las aristas B, por otro lado el *RPP* solo debe recorrer al menos una vez las aristas A.

Finalmente, el Problema de Ruteo de por Arcos con Capacidad limitada en los Vehículos (*Capacitated Arc Routing Problem o CARP*) fué sugerido por primera vez por Golden y Wong en 1981 [5], como una extensión del problema del Cartero Rural, con restricciones de capacidad en los vehículos.

El problema consiste en encontrar un conjunto de rutas para una flota de vehículos con capacidad de carga homogénea, de tal forma que satisfaga las demandas de un conjunto de calles (arcos) de una red vial, iniciando y terminan sus recorridos desde un único depósito [5], como se muestra en la figura 2.4.

El objetivo del problema es minimizar el costo total de recorrido de manera tal que se atiendan todas las demandas sin exceder la capacidad de carga de los vehículos. Cada una de las demandas asociadas al conjunto de calles debe ser atendida por un solo vehículo y a su vez ningún vehículo debe exceder su capacidad de carga.

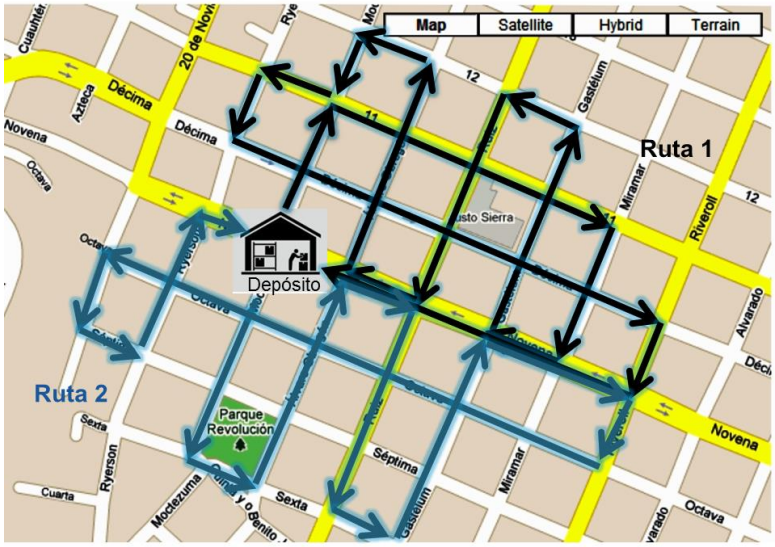


Figura 2.4: Representación gráfica de una solución factible de dos rutas para el CARP.

En el *CARP* existen 2 tipos de aristas: aristas requeridas y aristas no requeridas o de paso. Las aristas requeridos son aquellas que tienen asociada una demanda que debe ser satisfecha y por tanto deben ser visitados, mientras que las aristas no requeridas son visitados como medio de conexión entre otros dos aristas requeridas. Retomando la Figura 2.3, las aristas A representan las aristas requeridas mientras que las aristas B corresponden a las aristas no requeridas.

Si un grafo G es completamente dirigido o completamente no dirigido, este puede ser resuelto en un tiempo de computo rápido o factible, es decir, tiempo polinomico y se le considera un problema *NP*, por el contrario, si el grafo es mixto (dirigido y no dirigido), es considerado un problema *NP-Difícil* y su tiempo de solución es exponencial. Edmonds y Johnson demostraron que *CARP* es considerado *NP-Difícil* [15].

En la Figura 2.5 se muestra la diferencia entre el *CPP*, *RPP* y *CARP*. En resumen, los tres problemas consisten en encontrar un recorrido con el menor costo posible que comience y termine en el mismo punto, para *CARP* se requiere encontrar una cantidad de recorridos igual o menor a los vehículos disponibles, visitando solo un subconjunto de todos los arcos o aristas del grafo de la misma forma que el *RPP*.

Figura 2.5: El RPP es un CPP con la restricción de que debe visitar solamente un conjunto de calles, el CARP en un RPP con restricción de capacidad y de igual manera el CARP es un CPP con ambas restricciones (capacidad y un conjunto de calles a visitar).

2.2. Modelo matemático del CARP

En la literatura se han propuesto diferentes formulaciones matemáticas para el *CARP* tanto para el caso dirigido y no dirigido. El problema que pretendemos resolver, como se mencionó anteriormente, está inspirado en la recolección de residuos sólidos urbanos, el cual está caracterizado por el hecho de que la red vial de interés debe ser modelado como un grafo no dirigido, por lo que el problema a resolver es un *CARP* no dirigido.

A continuación se describe una formulación matemática propuesta por Maniezzo [16], donde:

$G = (V, E)$ es un grafo no dirigido que representa la red vial;

$|V|$ es el conjunto de vértices (nodos), donde el 0 corresponde al depósito;

$|E|$ es el conjunto de arcos, donde cada uno tiene asociado una pareja de valores (c_{ij}, q_{ij});

$R \subseteq E$ es el conjunto de arcos requeridos;

$V_r \subseteq V$ es el conjunto de vértices conteniendo los extremos de las aristas de R,

más el vértice asociado al depósito;

$K = (1, ..., M)$ es la flota de vehículos con capacidad de carga Q;

c_{ij} es el costo de recorrer del arco $(i, j) \in E$;

q_{ij} la demanda asociada al arco $(i, j) \in R$;

$$q_{ij} = \left\{ \begin{array}{ll} 0 & \text{si } (i,j) \notin R \\ > 0 & \text{si } (i,j) \in R \end{array} \right\}$$

Las variables de decisión son definidas como:

$$x_{ijk} = \left\{ \begin{array}{ll} 1 & \text{si el arco } (i,j) \text{ es recorrido por el vehículo } k \\ 0 & \text{en cual quier otro caso} \end{array} \right\}$$

El CARP puede ser modelado como un problema de programación lineal de la siguiente forma:

$$Minimizar = \sum_{(i,j)\in E} c_{ij} \sum_{k\in K} x_{ijk} \tag{2.1}$$

$$\sum_{k\in K} x_{ijk} = 1 \quad (i,j) \in R \tag{2.2}$$

$$\sum_{j\in V_r - \{0\}} \sum_{k\in K} x_{0jk} = |K| \tag{2.3}$$

$$\sum_{(i,j)\in R} q_{ij} x_{ijk} \leq Q \quad k \in K \tag{2.4}$$

$$\sum_{j\in V_r} x_{ijk} = \sum_{j\in V_r} x_{jik} \quad i \in V_r, k \in K \tag{2.5}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ijk} \geq \sum_{j\in V} x_{hjk} \quad S \subseteq V_r - \{0\}, k \in K, h \in S \tag{2.6}$$

$$x_{ijk} \in \{0,1\} \quad (i,j) \in R, k \in K \tag{2.7}$$

Función objetivo 2.1 busca minimizar el costo total de recorrido, sujeto a las siguientes restricciones:

- Asegura que todos los arcos requeridos deben ser atendidos por un solo vehículo 2.2;
- El número de vehículos a usar es igual al número de vehículos disponibles 2.3;
- Asegura que no se exceda la capacidad de los vehículos disponible 2.4;
- Da continuidad a las rutas, es decir, si el vehículo K entra a una calle debe salir de ella 2.5;
- Eliminación de sub-rutas, garantiza que no haya ciclos dentro de la ruta antes de regresar al depósito 2.6;
- Asegura que la variable desición tome solo valores de 0 y 1, es decir es binaria 2.7.

.

2.3. Métodos de solución para el CARP

Desde que el *CARP* fue sugerido en 1981 por Golden y Wong [5], en la literatura se han propuesto algoritmos exactos y aproximados para su solución. Los primeros, aunque nos proporcionan soluciones óptimas, el tamaño de las instancias resueltas son muy pequeñas comparadas con las instancias resueltas por los algoritmos aproximados.

A medida que el problema aumenta, crece también la cantidad de combinaciones para resolverlo y su tiempo de computó es demasiado alto, supongamos un problema de rutas donde se deben visitar 37 clientes, entonces se tienen 36!/2 combinaciones lo que da como resultado un valor de 1.86 x 10^{41} [17] que es un número grande de posibilidades para un número pequeño de clientes a visitar, como este y muchos problemas el crecimiento de las combinaciones es de tipo exponencial, por esto en muchos de los problemas reales se vuelve imposible usar métodos exactos para encontrar la solución óptima incluso para los procesadores de alto rendimiento. Otros ejemplo de este tipo de problemas combinatoriales y de solución no polinómica se muestra en el libro "Técnicas metaheurísticas de optimización"[18].

Dada la necesidad de encontrar soluciones en un tiempo computacional razonable para este tipo de problemas combinatorios, surgen las técnicas heurísticas y metaheurísticas (algoritmos aproximados), estos métodos exploran algunos espacios de soluciones que al parecer sean los que presenten mayor indicio de que exista una solución adecuada, lo que hace que el tiempo de búsqueda sea más bajo y práctico para un sistema computacional, generándose así una solución válida que se le conoce como óptimo local.

A continuación se describen los algoritmos propuestos por diferentes autores, los cuales se han clasificados en exactos, heurísticos y metaheurísticos.

2.3.1. Algoritmos exactos

Estos algoritmos son muy eficientes para problemas de ruteo con una cantidad reducida de clientes como lo menciona N. Azi et al. [19], en su artículo donde resuelve una variante del problema de rutas de vehículos (VRP) con ventanas de tiempo, afirma en sus conclusiones que los métodos clásicos de programación lineal entera (ILP) pueden solucionar de forma sencilla casos con 25 clientes y en algunos casos con un máximo de 50 clientes debido a restricciones de tiempo computacional.

Los algoritmos Branch and Bound (ramificación y acotamiento) y Cutting Plane (plano de corte) han sido aplicados para resolver el *CARP*, estos métodos están basados en *ILP*, es decir, parten de una descripción matemática formal del problema, donde se formula con una función objetivo, que consiste en minimizar o maximizar alguna función sobre algunas variables de decisión y un conjunto de restricciones que deben ser satisfechos en cualquier solución factible al problema.

El método Branch and Bound, consiste encontrar aquellas ramas del árbol de soluciones que ya no están siendo óptimas (infactibles o de baja calidad) para podarlas (Bound) y ramificar (Branch) aquellas que pueden contener las soluciones óptimas. Los primeros en resolver el *CARP* con este método fueron Hirabayashi [20] y Kiuchi [10], ambos autores resolvieron el problema con instancias de prueba con aristas requeridas que varían entre 15 y 50, en las cuales la soluciones óptimas fueron encontradas.

Por otro lado el método Cutting Plane a diferencia de resolver subproblemas para cada rama del árbol, este propone trabajar con un único problema (*ILP*) agregando un conjunto de restricciones (*Cuts*), siempre que estas aseguren que el problema cumpla con la función objetivo, esto se repite hasta que se encuentre (de ser posible) la solución óptima.

Belenguer y Benavent [21] ha propuesto un algoritmo basado en Cutting Plane para *CARP*, realizando pruebas con instancias (Gdb y Kshs) que van hasta 50 nodos y 97 aristas requeridas, donde se alcanza un GAP (brecha respecto a su solución y la mejor solución conocida) inferior al 1 %, e instancias (Eglese) que van hasta 140 nodos y 190 aristas requeridas, logrando un GAP menor al 3 %.

2.3.2. Algoritmos heurísticos

Los algoritmos heurísticos son técnicas que buscan buenas soluciones (no necesariamente la óptima) a problemas difíciles en un tiempo computacional razonable [1]. Aunque los resultados obtenidos por estos algoritmos no aseguran soluciones óptimas, estos se alcanzan con un costo computacional aceptable y de igual calidad como las obtenidas por métodos exactos.

Las heurísticas se clasifican en algoritmos de construcción (paso a paso va agregando de forma aleatoria o mediante alguna regla elementos a la solución parcial hasta obtener una completa) y algoritmos de búsqueda local (iterativamente realiza cambios a elementos de una solución con el fin de mejorarla).

Se han propuesto diferentes algoritmos heurísticos para la solución del *CARP*, alguno de ellos son:

Augment-Merge: propuesto por Golden y Wong [5], donde construye los recorridos mediante dos fases Aumentar y luego Unir. Construye tantas rutas como aristas requeridas, después trata de que cada una ruta absorba las aristas requeridas por otra, siempre que cumpla la capacidad de vehículo, al final intenta conectar dos rutas buscando que mejoren el costo de su recorrido

Path Scanning: sugerido por Golden y Baker [22], este algoritmo construye un recorrido a la vez agregando una nueva arista requerida al recorrido actual hasta que ya no se pueda agregar ninguna arista requerida debido a la capacidad del vehículo, esto se repite hasta que se hayan incluido todas las aristas requeridas en al menos una ruta.

Ulusoys Algorithm (Route-First Cluster-Second): formulado por Gunduz Ulusoy [23], donde se construye soluciones en dos fases. En la primera, se obtiene una única ruta que visita todos los arcos requeridos (no considera las restricciones de capacidad del vehículo). En la segunda, se agrupan los arcos requeridos en diferentes rutas, satisfaciendo la restricción de capacidad en cada una de ellas.

Shorten, Swich, Add, Drop y Cut: presentado por Hertz, Laporte y Mittaz [24], proponen varios algoritmos que trabajan en forma conjunta para obtener soluciones al problema de ruteo. El algoritmo regresa una ruta factible con todas las aristas requeridas sin tomar en cuenta la restricción de capacidad de los vehículos (similar a usuloy), después quita una arista requerida de alguna ruta dada, a la vez que agrega otra arista y cambia el sentido del recorrido, finalmente intenta reducir el costo de recorrido eliminando cadenas de aristas requeridas revisadas previamente.

Parallel Insert: introducido por Chapleau en 1984 [25], esta estrategia utiliza un procedimiento de inserción para generar muchas rutas en paralelo. El objetivo es obtener un mejor equilibrio entre los costos de transporte de cada ruta.

2-OPT y 3-OPT: propuesto por Croes (1958) [26], dada una solución inicial, elimina 2 aristas y las conecta a una misma o diferente ruta, el algoritmo 3-OPT elimina 3 aristas y las reconecta a otro ruta.

2.3.3. Algoritmos metaheurísticos

Por otro lado, problemas *CARP* se han tratado con metaheurísticas, que son procedimientos de búsqueda, que al igual que los heurísticos tampoco garantizan la obtención de la solución óptima del problema considerado y se basan en la aplicación de reglas relativamente sencillas.

A diferencia de los heurísticos, las técnicas metaheurísticas tratan de evitar óptimos locales orientando la búsqueda en cada momento dependiendo de la evolución del proceso de búsqueda. Las metaheurísticas producen en general mejores soluciones.

En base a sus características, la metaheurística se pueden clasificar [9]:

- Según su origen, estos algoritmos están basada en su fuente de inspiración (por ejemplos inspirados en la naturaleza).

- Según el número de soluciones utilizadas al mismo tiempo, en esta clasificación existen dos categorías, los algoritmos basados en poblaciones que trabajan con un conjunto de soluciones en el proceso de búsqueda y los basados en métodos de trayectoria que trabajan soluciones únicas durante el proceso de búsqueda.

- Según como hacen uso de la función objetivo, mientras que algunos algoritmos mantienen la función objetivo dada en la representación del problema "tal como es", algunos otros, la modifican durante la búsqueda. La idea detrás de este enfoque es escapar de mínimos locales modificando el espacio de búsqueda. En consecuencia, durante la búsqueda la función objetivo se altera intentando incorporar la información recolectada durante el proceso de búsqueda.

- Según la cantidad de estructuras de vecindario, la mayoría de los algoritmos metaheurísticos funcionan en una sola estructura de vecindad. En otras palabras, la topología del espacio no cambia en el transcurso del algoritmo. Otras metaheurísticas, utilizan un conjunto de estructuras vecinales que ofrece la posibilidad de diversificar la búsqueda intercambiando entre diferentes espacios de la vecindad.

- Según su uso o no de memoria, los algoritmos sin memoria utilizan la información exclusivamente para determinar la siguiente acción es el estado actual del proceso de búsqueda, por otro lado los algoritmos con memoria se diferencian entre el uso de la memoria a corto y largo plazo.

- Según la forma en que construyen la solución o exploran el vecindario de una solución, la cual puede ser de manera aleatoria o determinística.

A continuación en la Tabla 2.1, se muestran diferentes metaheurísticas que se han modelado para resolver el *CARP* desde el año 2000 a la fecha, las cuales se resumen en las siguientes:

Tabu Search (TS): introducido por Glover [27] y aplicado para resolver el *CARP* por Hert, Laporte y Mittaz [28] es un algoritmo que se caracteriza porque utiliza una estrategia basada en el uso de estructuras de memoria para escapar de los óptimos locales, en los que se puede caer al moverse de una solución a otra por el espacio de soluciones.

Variable Neighborhood Descent (VND): introducido por Hertz y Mittaz [29] es un método determinístico. La idea es buscar sistemáticamente en diferentes esquemas de vecindad hasta llegar a un mínimo local (mínimo con respecto a todos los esquemas de vecindad).

Guide local search (GLS): introducido por primera vez por C. Voudoris [30] es un algoritmo que ayuda a otro algoritmo de búsqueda local a escapar de un óptimo local, modificando el espacio de búsqueda a través de penalizaciones.

Ant Colony (AC): desarrollado por Lacomme [31] es un algoritmo inspirado en el comportamiento de colonia de hormigas de como buscan su alimento.

Genetic Algorithm (GA): introducido a este tipo de problemas por Deng [32] es un algoritmos inspirados en la evolución biológica y en su base genético-molecular.

Memetic Algorithm (MA): propuesto por Moscato [33] este algoritmo se basa en la evolución de poblaciones, el objetivo es hacer que la búsqueda local y el algoritmo trabajen de forma cooperativa que permita mejorar el proceso de búsqueda.

GRASP : fue propuesto por Feo y Resende en 1989 [34] es un algoritmo de búsqueda basado en una función voraz, aleatoria y adaptativa.

Biased Random-Key Genetic Algorithms (BRKGA): propuesto por Resende y Goncalves en 2011 [35] y aplicado al *CARP* por Martinez [36] este algoritmo, un decodificador genera claves aleatorias, toma como entrada cualquier cromosoma y asocia con él una solución del problema de optimización combinatoria para el cual se puede calcular un valor objetivo, el decodificador ordena el vector de claves aleatorias y los utiliza para representar una secuencia.

Año	Autor	Titulo	Metaheurística
2000	A Hertz [28]	A Tabu search heuristic for the capacitated arc routing problem	TS
2001	A Hertz [29]	A variable neighborhood descent algorithm for the undirected capacitated arc routing problem	VND
2003	P Beullens [37]	A guided local search heuristic for the capacitated arc routing problem	GLS
2004	KF Doerner [38]	Applying ant colony optimization to the capacitated arc routing problem	AC
2004	P Lacomme [31]	First competitive ant colony scheme for the CARP	AC
2007	X Deng [32]	A genetic algorithm for the capacitated arc routing problem	GA
2007	M Reghioui [39]	GRASP with path relinking for the capacitated arc routing problem with time windows	GRASP
2008	N Labadi [40]	GRASP with path relinking for the capacitated arc routing problem with time windows	GRASP
2008	J Brandão [41]	A deterministic tabu search algorithm for the capacitated arc routing problem	TS
2008	J Bautista [42]	Solving an urban waste collection problem using ants heuristics	AC
2009	K Tang [43]	Memetic algorithm with extended neighborhood search for capacitated arc routing problems	MA
2009	Y Mei [44]	Improved memetic algorithm for capacitated arc routing problem	MA
2009	M Morgan [45]	A weight-coded genetic algorithm for the capacitated arc routing problem	GA
2010	L Feng [46]	Towards probabilistic memetic algorithm: an initial study on capacitated arc routing problem	MA
2010	H Fu [47]	Memetic algorithm with heuristic candidate list strategy for capacitated arc routing problem	MA
2010	L Santos [48]	An improved ant colony optimization based algorithm for the capacitated arc routing problem	AC
2011	X Chen [49]	MAENS+: a divide-and-conquer based memetic algorithm for capacitated arc routing problem	AC
2011	C Martinez [36]	BRKGA algorithm for the capacitated arc routing problem	BRKGA
2013	T Liu [50]	A memetic algorithm with iterated local search for the capacitated arc routing problem.	MA
2013	FL Usberti [51]	GRASP with evolutionary path-relinking for the capacitated arc routing problem	GRASP
2014	M Liu [52]	Application specific instance generator and a memetic algorithm for capacitated arc routing problems	MA
2015	Z Wang [53]	Rank-based memetic algorithm for capacitated arc routing problems	MA
2016	Y Chen [54]	A hybrid metaheuristic approach for the capacitated arc routing problem	TS y ME

Tabla 2.1: Metaheurísticas de solución propuestas para el CARP.

2.4. Variaciones del CARP

En la tabla 2.2 se describen variantes del CARP que han sido propuestos como una forma de modelarlo con situaciones de la vida real.
Es importante aclarar que aunque existen diferentes variaciones al *CARP*, el desarrollo del presente trabajo de investigación aborda el problema clásico del *CARP*, es decir no incluye ninguna de las variaciones mencionadas en la tabla 2.2.

Sigla	Tipo de problema	Descripción
CARP-IF Ghiani [55]	CARP con instalaciones intermedias.	Este problema incluye instalaciones intermedias que son utilizadas para descargar las demandas de las aristas requeridas.
MD-CARP Amberg [56]	CARP con multi depósitos.	Se define como un CARP con varios depósitos (también conocido como 1-CARP) donde los vehículos son estacionados y desde donde los mismos inician y finalizan sus recorridos.
P-CARP Chu [57]	CARP periódico.	Este problema ademas de las características conocidas del CARP, considera un horizonte de periodos (por ejemplo días), cada arista requerida tiene una frecuencia que representa el número de servicios requeridos.
CARP-TW Reghioui [39]	CARP con ventanas de tiempo.	Se agrega la condición que la demanda de cada arista requerida sea atendida dentro de un intervalo de tiempo específico.
O-CARP Usberti [58]	CARP abierto.	En esta variante del CARP los vehículos que recorren las rutas no regresan al depósito.

Tabla 2.2: Variantes del CARP.

Entre las variadas metaheurísticas presentadas en la tabla 2.1 para la solución del problema planteado en la tesis, se ha seleccionado el *GRASP*, por su facilidad y sencillez en los procedimientos constructivos y de búsqueda local, ademas que el algoritmo ha demostrado ser eficiente para dar soluciones iniciales en la etapa de construcción. En la siguiente sección se describe dicha metaheurística.

2.5. Algoritmo GRASP

Greedy Randomized Adaptive Search Procedure (GRASP su sigla en inglés) es un algoritmo metaheurístico que resuelve problemas difíciles en el área de optimización combinatoria, fué diseñado por Feo y Resende en 1989 [34].

El GRASP es un procedimiento de búsqueda (Search) que no necesariamente evalúan a todos los elementos del problema, son voraces (Greedy) ya que eligen a los mejores elementos candidatos (elementos que pueden ser seleccionados para formar parte de una solución parcial) que cumplan ciertas propiedades, son adaptativos (Adaptive) porque cada generación de candidatos puede variar de acuerdo a las condiciones del problema y son aleatorias (Randomized) ya que los elemento candidatos son seleccionados aleatoriamente de una lista [59] y [60].

El GRASP es un proceso iterativo, en donde cada iteración se desarrollan dos etapas, en la primera se construye una solución y después en la segunda se mejora mediante algoritmos de búsqueda local. Cada vez que se ejecutan las dos etapas, la solución obtenida se almacena y se procede a realizar una nueva iteración, guardando cada vez la mejor solución que se haya encontrado hasta el momento. Un esquema general del algoritmo GRASP se presenta en el Algoritmo 1.

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.

A continuación se describen las etapas del GRASP [34]:

Etapa de construcción, consiste en construir iterativamente una solución factible incorporando un elemento candidato cada vez. Cada elemento candidato que se va agregando a la solución es determinada por la evaluación de todos los elementos en base a una función voraz, dicha función mide el beneficio o costo de agregar algún elemento candidato a la solución que se construye. Con la evaluación de los elementos se construye una lista restringida de candidatos (LRC) formada por los mejores elementos, por ejemplo, aquellos que al incluirse a la solución parcial genera un menor incremento

en el costo (este es el aspecto voraz del algoritmo). El siguiente elemento candidato a ser incorporado en la solución es seleccionado aleatoriamente de la LRC (componente de aleatoriedad).

Una forma de generar LRC es mediante cardinalidad: el elemento candidato se elige de los primeros k elementos, k a denir. Un enfoque diferente consiste en restringirse a aquellos elementos cuyos valores respecto a la función voraz se encuentran dentro de un rango. Este rango se controla a través de un parámetro $\alpha \in [0, 1]$. En ocasiones se aplican ambos enfoques en conjunto, la LRC se forma con los primeros k elementos y que estén dentro del rango definido.

Con un α=0 se vuelve un algoritmo completamente voraz, mientras que con un α=1 se vuelve aleatorio, en general α sirve para establecer un balance entre costo computacional y calidad de las soluciones.

Finalmente, una vez que se ha incorporado este elemento en la solución parcial, la LRC se actualiza y los elementos candidatos se vuelven a evaluar respecto a la función voraz; el hecho de que el beneficio o costo de los elementos se vea afectada por la última acción realizada refleja la parte de adaptatividad del algoritmo.

Etapa de mejora, el objetivo de esta etapa es mejorar la solución generada en la etapa de construcción, cualquier mecanismo que mejore la solución puede utilizarse, sin embargo, el uso de algoritmos de búsqueda en vecindarios (búsqueda local) son las más comunes para este propósito. Esta mejora se realiza de forma iterativa reemplazando sucesivamente la solución actual por otra que este en el conjunto de soluciones vecinas, esta selección se puede hacer de dos tipos:

Estrategia *Best improvement*: se busca la mejor solución entre el conjunto de soluciones vecinas y la solución actual es reemplazada por la solución del mejor vecino.

Estrategia *First improvement*: se toma la primera solución entre el conjunto de soluciones vecinas y se reemplazada por la solución actual (siempre y cuando sea mejor).

Esta fase termina cuando no encuentra una mejor solución en el conjunto de soluciones vecinas.

En la literatura, la metaheurística GRASP se ha utilizado para resolver gran variedad de problemas de optimización combinatoria, específicamentelos algunos trabajos publicados que resuelven problemas de rutas de vehículos con este algoritmo se muestran en la tabla 2.3.

Nombre del articulo	**Autor**
Clustered prize-collecting arc routing problem	Aráoz (2013)
A GRASP algorithm based on new randomized heuristic for vehicle routing problem	Layeb (2013)
GRASP with evolutionary path-relinking for the CARP	FL Usberti (2013)
Multiple phase neighborhood search-GRASP for the capacitated vehicle routing problem	Marinakis (2012)
Solving the capacitated vehicle routing problem and the split delivery using GRASP metaheuristic	Suárez (2012)
Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem	Salehipour (2011)
GRASP with path relinking for the CARP with time windows	N Labadi (2008)
Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking	Prins (2006)
A fast GRASP with path relinking for the capacitated arc routing problem	Prins C (2005)
Expanding neighborhood GRASP for the traveling salesman problem	Marinakis (2005)

Tabla 2.3: Problemas resueltos por GRASP encontrados en la literatura.

Julián Aráoz [61] aplica GRASP y lo complementa con el algoritmo Path Relinking, Layeb [62] utiliza el GRASP con una nueva forma de aleatorizar, Marinakis [63] resuelve el problema con GRASP que incluye un algoritmo de Neighborhood Search con multiple fase, Salehipour [64] por su parte incluye los algoritmos VND y VNS, Suárez [65] aplica el clásico GRASP sin ninguna variante, Prins [66] complementa el GRASP con el algoritmo path relinking y Marinakis [67] agrega una expanción al vecindario en la fase de mejora del GRASP.

Por otro lado, en la misma tabla 2.3 se observa que en solo 3 artículos la metaheurística *GRASP* fué aplicado para dar solución al CARP:

Labadi [40] aplican el *GRASP* con Path Realinking para resolver el CARP con la variante de ventanas de tiempo (*CARP-TW*), es decir que cada arista requerida ademas de ser cubierta su demanda debe ser atendida dentro de un intervalo de tiempo por los que los hace diferente al modelado en este trabajo.
Prins [68] y Usberti [51], resuelve el CARP con la metaheurística del presente trabajo, la diferencia radica en la forma como se aplica el algoritmo *GRASP*, ellos agrega el algortimo Path-Relinkig en la fase de mejora como complemento, ademas en [51], en

la etapa de construcción del algoritmo, incluye un proceso de ajuste de parámetros, basado en el trabajo de [69].

Con base a los resultados mostrados en la literatura 2.3 y partiendo de la premisa de que soluciones iníciales diversas y de buena calidad juegan un papel importante en el éxito de métodos de búsqueda locales óptimas en regiones prometedoras del espacio de soluciones [70], el algoritmo *GRASP* resulta ser una alternativa para la solución del problema del *CARP*.

Asimismo, el *GRASP*, debido a su simplicidad, es generalmente muy rápido y es capaz de producir muy buenas soluciones en un tiempo muy corto de computación. Además, puede integrarse con éxito en otras técnicas de búsqueda [9], lo cual resulta bastante atractivo para el presente trabajo, ya que uno de los trabajos futuros es integrar otros operadores de mejora basados en algoritmos genéticos y *BRKGA*.

Por lo anterior, se selecciona el algoritmo *GRASP*, en donde se incluyen operadores de mejora diferentes a los presentados en la literatura 2.3, igualmente se realiza una calibración de parámetros como una etapa de pre-procesamiento.

Capítulo 3

Desarrollo de la investigación

En este capítulo se describe la metodología DIMMA, posteriormente se muestra el algoritmo propuesto para la solución del CARP mediante el algoritmo GRASP. Se explican los operadores de búsqueda local que se proponen en este trabajo en la etapa de mejora del GRASP.

3.1. Metodología DIMMA (*Design and Implementation Methodology for Metaheuristic Algorithms*)

La metodología utilizada para el diseño e implementación de la metaheuristica aplicada en esta investigación es la que propone Yaghini (2012) [71] llamada DIMMA.
DIMMA consta de tres fases secuenciales que a la vez cada una de ellas tiene varios pasos como se muestra en la Figura 3.1.
En cada paso, se definen varias actividades, que deben realizarse hasta completar todas las fases de la metodología. Estas fases son las siguientes: iniciación, en la que el problema debe ser entendido con precisión así como el objetivo de diseñar la metaheurística también debe ser claramente definido en esta fase.
La siguiente fase es el planteamiento, los objetivos más importantes de esta fase son seleccionar el método de solución metaheurístico, definir medidas de desempeño y diseñar algoritmos para la estrategia de solución.
La última fase es la construcción en la que se debe implementar el algoritmo diseñado, ajuste de parámetros de entrada (parametrización), analizar su desempeño y finalizar con la documentación de los resultados.
En algunos pasos, es necesario revisar los pasos anteriores para justificar y mejorar las decisiones en el algoritmo. Por ejemplo, es común que el algoritmo sea modificado después de las evaluaciones de rendimiento. Estos movimientos hacia atrás se ilustran en la Figura 3.1.

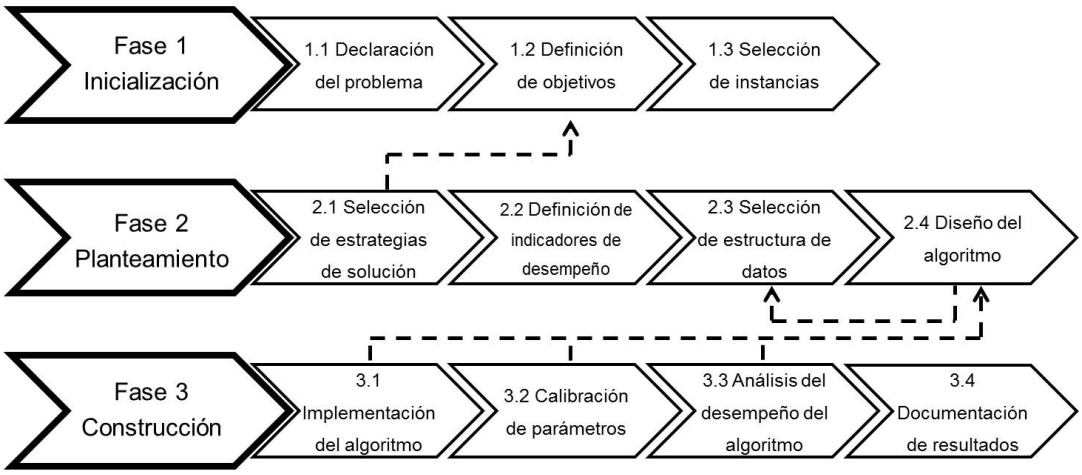


Figura 3.1: Fases de la Metodología DIMMA.

A continuación se describen los pasos para cada fase de DIMMA:

- Paso 1.1: Declaración del problema. El primer paso es definir y delimitar el problema. Para expresar el problema, se puede escribir un enunciado simple que incluya uno o más objetivos, insumos, productos y supuestos de problema.
  En este paso, cuando sea posible, se puede proporcionar un modelo matemático para mayor claridad.
  Se debe definir como estará estructurado el problema, los enfoques multi-objetivos, los aspectos dinámicos, el modelado continuo o discreto, la definición de objetivos, la selección de estrategias de solución y la definición de medidas de desempeño, el definir bien estos aspectos será de gran importancia para los siguientes pasos.

- Paso 1.2: Definición de objetivos. Se deben definir claramente los objetivos del diseño de la metaheurística, ya que de esto dependerá los experimentos, medidas de análisis de desempeño y análisis estadístico.
  Algunos objetivos del diseño de la metaheurística pueden ser: (1) reducir el tiempo de búsqueda en comparación con métodos exactos u otras metaheurísticas, (2) mejorar la calidad de las soluciones, (3) robustez en términos de instancias, (4) resolver problemas a gran escala, (5) facilidad de implementación, (6) facilidad para combinar con otros algoritmos para mejorar el rendimiento, (7) flexibilidad para resolver otros problemas o modelos de optimización, y (8) proporcionar una aproximación estrecha al problema.
  La selección de instancias, el método de solución y la definición de medidas de rendimiento deben realizarse de acuerdo con las objetivos seleccionados.

- Paso 1.3: Selección de instancias de prueba. En este paso se seleccionan las instancias que serán utilizadas para evaluar el algoritmo. Estas deben ser representativos al problema de estudio, además, para obtener un resultado interesante y permitir la generalización de las conclusiones, las instancias seleccionadas deben ser diversas en términos de tamaño de las instancias, su complejidad y su estructura, ya que esto será de gran utilidad en el análisis del rendimiento del algoritmo metaheurístico construido. Las instancias pueden dividirse en instancias reales y construidas. Las primeras son tomadas de ejemplos prácticos de aplicaciones del mundo real, las segundas son las que encontramos en la literatura y son utilizadas con más frecuencia en las experimentaciones debido a que se puede comparar con otros métodos en la literatura.
  Después de seleccionar instancias, estas 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.

- Paso 2.1: Selección de estrategias de solución. Una vez de revisados los métodos de solución existentes para el problema, debe justificarse la necesidad de utilizar metaheurísticas. De hecho, de acuerdo con los métodos de solución existentes, debe distinguirse si la aplicación metaheurística para el problema es necesaria o no. Si se selecciona un enfoque metaheurístico como método de solución, se puede pasar al siguiente paso; de lo contrario debemos detenernos en este punto.
  De acuerdo a la situación, podemos seleccionar una de las estrategias de solución como algoritmo exacto, algoritmo heurístico, algoritmo metaheurístico, método híbrido, algoritmo paralelo y algoritmo cooperativo.
  Una vez seleccionada la estrategia de la solución, debemos especificar los componentes del algoritmo para nuestro problema. Esta especificación puede ser útil en el paso de seleccionar la estructura de datos y diseñar el algoritmo.

- Paso 2.2: Definición de indicadores de desempeño. En este paso, se define como se medirá el rendimiento del algoritmo. Debemos seleccionar las medidas apropiadas de acuerdo con los objetivos seleccionados en el paso 1.2. 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 las soluciones se basa en la medición de la distancia (GAP) a una de las siguientes soluciones mostradas en la Figura 3.2 : solución óptima, solución inferior (superior), solución mejor conocida y solución real implementada. Para las instancias construidas, el óptimo es conocido. En este caso, el resultado del porcentaje de las corridas es equivalente al óptimo global. Sin embargo, normalmente este óptimo global no es conocido, por lo que el límite inferior o superior se puede utilizar para problemas de minimización y maximización, respectivamente.

Para algunos problemas estándar y populares, la mejor solución conocida está disponible en la literatura que puede usarse como un indicador para evaluar la calidad de las soluciones. Para la aplicación en el mundo real, puede haber un objetivo predefinido que puede ser una medida de calidad y robustez.

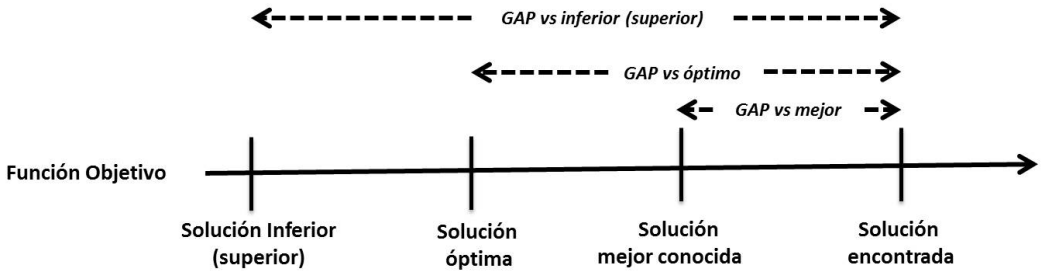


Figura 3.2: Evaluación del desempeño respecto a la calidad de la soluciones para un problema de minimización

- Paso 2.3: Selección de estructura de datos. Antes de diseñar el algoritmo, es necesario seleccionar una estructura de datos adecuada en este paso. La estructura de datos es un esquema para organizar la información en la memoria, para una mejor eficiencia de algoritmos. Los arreglos, archivos, listas, matrices, árboles y tablas son los tipos importantes de estructuras de datos. Seleccionar la estructura de datos correcta puede hacer una enorme diferencia en la complejidad de la implementación resultante.

- Paso 2.4: Diseño del algoritmo. Aquí se debe especificar la estructura global del algoritmo. Se refiere a la forma como se específica el programa, la cual puede ser en su forma más sencilla (lenguaje natural). Pseudocódigo es otra forma de especificar un algoritmo que es una combinación de lenguaje natural y de programación. Utilizando pseudocódigo, se puede especificar el algoritmo con mayor precisión que un lenguaje natural. En lugar de dos formas anteriores para especificar un algoritmo, se puede utilizar el diagrama de flujo.

  El algoritmo también debe ser analizado 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 (entre más simple es el algoritmo, más fácil puede ser programado y depurado y por último la generalidad (es la aplicabilidad de un algoritmo a una amplia gama de insumos).

- Paso 3.1: Implementación del algoritmo. En esta fase el algoritmo es implementado mediante un lenguaje de programación adecuado. Por ejemplo, si un algoritmo

consiste en objetos y métodos relacionados, entonces será mejor implementar dicho algoritmo usando uno de los lenguajes de programación orientados a objetos tales como C ++, C o Java.

En este paso, con el fin de acelerar la programación y reducir los costos de desarrollo, se puede utilizar frameworks de software libre. Los frameworks de software son códigos de programación reutilizables y componentes para la metaheurística.

- Paso 3.2: Ajuste de parámetros.En este paso se debe hacerse la sintonización de parámetros, también conocida como ajuste de parámetros. Para ello, hay que afinar varios números de parámetros numéricos y/ o categóricos, y para ello puede y debe utilizarse el método científico y el análisis estadístico.

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

  Existen dos estrategias diferentes para el ajuste de parámetros:
  La primera es el ajuste offline, los parámetros se establecen antes de la ejecución de las metaheurísticas y no se actualizan durante la ejecución. En esta estrategia, la sintonización de parámetros uno a uno no garantiza la optimalidad de los valores de los parámetros, ya que no hay interacción entre los parámetros. Para superar este problema, se puede utilizar el Diseño de Experimentos (DOE).

  DOE se refiere al proceso de planificación del experimento para que los datos apropiados que pueden ser analizados por métodos estadísticos serán recolectados, 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.

  La segunda estrategia es el ajuste online, que actualizan 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, cambiando la solución, el valor de los parámetros también se cambia.

- Paso 3.3: Análisis del desempeño del algoritmo. En este paso, debemos obtener los resultados experimentales para diferentes indicadores para analizar el desempeño de algoritmos metaheurísticos con pruebas estadísticas, tales como test t, y

modelos ANOVA para una comparación de más de dos algoritmos.
Se deben llevar a cabo muchos ensayos (al menos 10, más de 100 si es posible) para obtener resultados estadísticos significativos. De este conjunto de ensayos se pueden calcular muchas mediciones Talbi (2009) [72] como la media, la mediana, el mínimo, el máximo, la desviación estándar, el porcentaje de éxito (%GAP) de la solución de referencia (la solución óptima o solución mejor conocida).

- Paso 3.4: Documentación de resultados. En este paso de DIMMA, los resultados deben ser analizados e interpretados basados en los objetivos planteados en el paso 1.2 y los indicadores de desempeño definidos. Generalmente, las herramientas gráficas son más comprensibles que la presentación de la gran cantidad de resultados de datos utilizando tablas. Las gráficas de interacción representan la interacción entre diferentes factores y su efecto sobre la respuesta obtenida (medida de rendimiento).

3.2. Algoritmo del GRASP aplicado al CARP

En esta sección se detallan los pasos del algoritmo planteado en la sección 2.5 del capítulo 2 con el fin de cumplir con el objetivo general.

3.2.1. Datos de entrada al programa

Primeramente, para comparar los resultados obtenidos por este trabajo, se seleccionaron las instancias de prueba propuestas por M. Kiuchi (KSHS) [10] que tiene hasta 10 nodos y 15 aristas, B.L. Golden (GDB) [5] que comprende hasta 27 nodos y 55 aristas y E. Benavent (VAL) [11] que contiene hasta 50 nodos y 97 aristas, dichas instancias también fueron comparadas con los resultados obtenidos en los modelos propuestos por la literatura mostrados en las tablas 2.1 y 2.3.

La figura 3.3 muestra un ejemplo de una instancia de prueba. Cada instancia es un archivo con extención "*.dat*" que consta de varios campos, los cuales se describen al la Tabla 3.1.

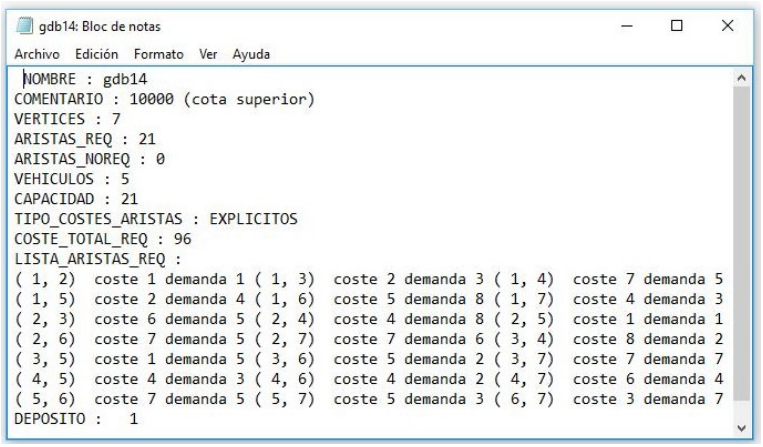


Figura 3.3: Ejemplo de instancia de prueba de 7 nodos y 21 aristas requeridas.

Contenido	**Descripción**
NOMBRE	Es el nombre de la instancia y es utilizado para identificar el archivo
COMENTARIO	Es un comentario adicional a los datos (normalmente se pone un limite superior).
VERTICES	Especifica el número de vértices (nodos)
ARISTAS REQ	Establece el número de aristas con demanda positiva
ARISTAS NOREQ	Especifica el número de aristas con demanda cero, es decir aristas no requeridas.
VEHICULOS	Se refiere al número de vehículos disponibles
CAPACIDAD	Proporciona la capacidad de carga del vehículo
TIPO COSTES ARISTAS	Indica el tipo de costo que utiliza las aristas
COSTO TOTAL REQ	Es la suma de los costos de todas las aristas requeridas
LISTA ARISTAS REQ	En esta sección se en listan las aristas con demanda positiva. Los número entre paréntesis es el par de nodos correspondientes a los extremos de cada arista. Para cada una de las aristas se especifica su costo de traslado y su demanda respectivamente.
DEPOSITO	Se refiere al vértice (nodo) que representa al deposito (generalmente este corresponde al nodo 1)

Tabla 3.1: Contenido de instancias de prueba, la primer columna muestra los campos contenidos en cada instancia y la segunda columna describe a que se refiere el campo correspondiente.

El programa inicia con la lectura de datos la instancia, con esta información genera una matriz de costos de distancias (desde el nodo i hacia el nodo j) de tamaño n x n, donde n= número de nodos, la Figura 3.4 muestra una ejemplo de una matriz de 8 x 8. En esta matriz nos dice que de ir desde nodo 1 (depósito) hacia el nodo 2 tiene un costo de tranporte de 1247, ir del nodo 1 al nodo 3 cuesta 737, ir del nodo 4 hacia nodo 1 su costo es de 1400 y así sucesivamente.

	1	2	3	4	5	6	7	8
1	0	1247	737	1400	1321	1484	1048	1082
2	1247	0	510	166	745	780	370	984
3	737	510	0	663	935	801	311	696
4	1400	166	663	0	911	946	468	1046
5	1321	745	935	911	0	641	1115	239
6	1484	780	801	946	641	0	490	402
7	1048	370	311	468	1115	490	0	892
8	1082	984	696	1046	239	402	892	0

Figura 3.4: Matriz de costos de traslados generada por el programa de tamaño 8 x 8.

Posteriormente genera una segunda matriz con las demandas de servicio asociada a la arista (i, j) de tamaño igual a la anterior. La figura 3.5 se muestra un ejemplo donde la demanda es la recolección de basura en kilogramos, es decir la arista $(1, 2)$ demanda un servicio de 24 kg, la arista $(2, 3)$ demanda un servicio de 6 kg, la arista $(2, 5)$ tiene una demanda de 0 kg (arista no requerida), y así sucesivamente.

	1	2	3	4	5	6	7	8
1	0	24	4	7	4	13	10	1
2	24	0	6	5	0	0	0	0
3	4	6	0	3	5	0	0	0
4	7	5	3	0	1	4	8	1
5	4	0	5	1	0	6	0	0
6	13	0	0	4	6	0	4	2
7	10	0	0	8	0	4	0	3
8	1	0	0	1	0	2	3	0

Figura 3.5: Matriz de demandas de servicios de tamaño 8 x 8.

Una vez generadas las matrices de costos de distancias y demandas requeridas, mediante el algoritmo Floyd-Warshall propuesto por Floyd [73] son creadas dos matrices adicionales:

Matriz de menor costo de traslado, muestra el camino con el menor costo de traslado desde un nodo i hacia un nodo j, además se asignan valores de 9999 (costo) para nodos que no tienen conección y valores de 0 cuando el nodo i = nodo j.
Un ejemplo de esta matriz se muestra en la figura 3.6, donde se observa que el camino con menor costo de traslado desde el nodo *1* hacia *3* es de 737, por otro lado, se muestra que no hay conección para ir desde el nodo *2* hacia el *8*, debido a que tiene un valor asignado de 9999.

Matriz de precedencias, muestra el camino que se debe recorrer para llegar desde un nodo i hacia un nodo j. Un ejemplo se encuentra en la figura 3.7, donde se observa que para llegar desde el nodo *1* al *2* se hace directamente, es decir no existen nodos entre ellos la ruta seria $1-->2$, en cambio para llegar desde nodo *1* hacia el nodo *7*, el camino a seguir seria del *1* al *2* después al *6* y finalmente al *7*, la ruta sería $1-->2-->6-->7$.
En el algoritmo 2 se muestra el pseudocódigo que genera las matrices de los caminos con menor costo de traslado y de precedencias.

Algoritmo 2: Floyd-Warshall

Entrada: n= número de vértices, C= Matriz de distancias
Salida : P= Matriz de menor costo de traslado, B= Matriz de precedencias

```
for i ← 1 to n do
    for j ← 1 to n do
        if i=j then
            P_ij = 0; B_ij = 0
        else P_ij = 9999; B_ij = j ;
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            suma = C_ik + C_kj
        if suma <C_ij then
            P_ij = suma; B_ij = B_ik
return Matriz P y Matriz B;
```

	1	2	3	4	5	6	7	8
1	0	9999	737	9999	9999	9999	9999	1082
2	9999	0	510	166	745	780	370	9999
3	737	510	0	663	9999	9999	311	696
4	9999	166	663	0	9999	9999	468	1046
5	9999	745	9999	9999	0	9999	9999	239
6	9999	780	9999	9999	9999	0	490	402
7	9999	370	311	468	9999	490	0	9999
8	1082	9999	696	1046	239	402	9999	0

Figura 3.6: Ejemplo matriz del camino con menor costo de traslado.

	1	2	3	4	5	6	7	8
1	0	2	2	2	7	7	2	7
2	2	0	2	3	4	5	6	4
3	0	1	0	3	7	6	6	7
4	2	1	2	0	1	1	6	7
5	7	1	7	1	0	7	1	7
6	7	1	6	1	7	0	6	7
7	2	1	2	3	1	5	0	5
8	0	4	2	3	4	5	5	0

Figura 3.7: Ejemplo matriz de precedencias de tamaño 8 x 8.

El algoritmo *GRASP* esta planteado de la misma manera que el algoritmo general (Ver en la sección 2.5. A continuación se detallan las dos fases del algorimo que resolvera el *CARP*.

3.2.2. Etapa de construcción del GRASP aplicada al CARP

En esta etapa la solución inicial es construida tomando como base en el algoritmo ***Path Scanning*** (descrito en la sección 2.3.2), para este proyecto de investigación, el procedimiento comienza seleccionando la primer arista que formara parte de la ruta, esta arista es escogida aleatoriamente de la LRC y la conecta al depósito (nodo *1*) mediante una función voraz que consiste en evaluar cual de los dos extremos (i, j) de la arista seleccionada tiene un costo de recorrido menor con respecto al depósito, esta función define la orientación del arco, es decir si el costo de recorrido es menor desde el nodo *1* hacia nodo i, el dirección del arco sera $i-->j$ y el camino quedaría $1-->i-->j$, en caso contrario la dirección del arco sera $j-->i$ y el camino quedara $1-->j-->i$. Es importante aclarar que si existen aristas requeridas en el camino entre el depósito y el nodo seleccionado i o j, esta es seleccionada directamente y será atendida siempre que cumpla con la restricción de capacidad del vehículo. La Figura 3.8 muestra una representación gráfica de una solución parcial.

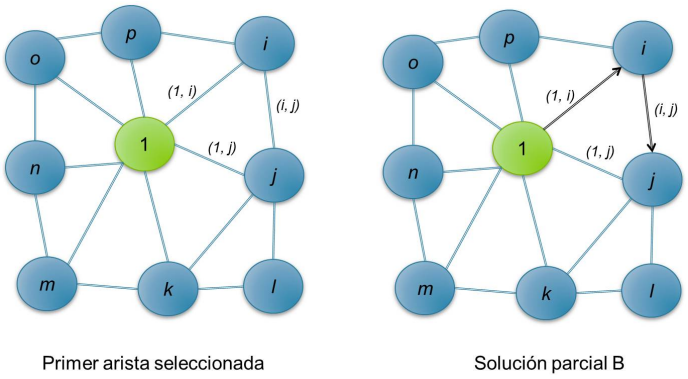


Figura 3.8: Primer arista selccionada es (i, j), con base a la matriz del camino con menor costo de traslado el primer arco es $1 - - > i$, luego $i - - > j$ por lo tanto la solución parcial B queda como se muestra $1 - - > i - - > j$.

Las siguientes aristas seleccionadas son evaluadas igualmente con la función voraz pero con respecto al último nodo de la ruta parcial construida hasta este momento, por ejemplo, si la solución parcial es $1 - - > 3 - - > 6$, la evaluación de la arista seleccionada (i, j) sera respecto al nodo 6 y asi sucesivamente con el resto de las aristas seleccionadas. Esto se repite hasta que la capacidad de carga del vehículo se termine o no exista ningúna arista requerido cuya demanda asociada sea menor al remanente de carga disponible del vehículo. La Figura 3.9 muestra una representación gráfica de una solución parcial con dos aristas seleccionadas.

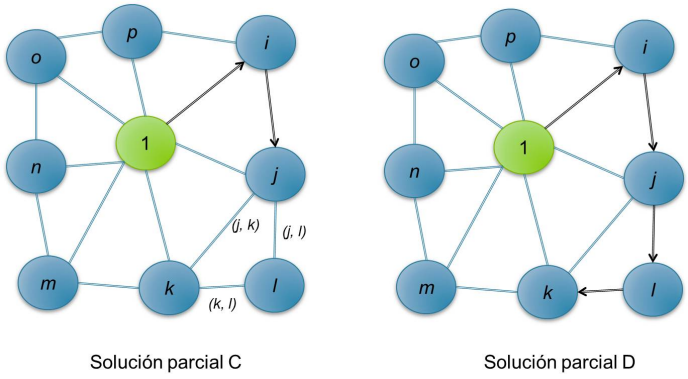


Figura 3.9: Segunda arista selccionada es (k, l), se evalúa el camino con menor costo de traslado desde el nodo j hacia el nodo k y l, como el menor costo es $j - - > l$ la solución parcial D queda como se muestra $1 - - > i - - > j - - > l - - > k$.

Finalmente el último nodo de la ruta hasta aquí construida se conecta al depósito por el camino de menor costo de traslado ofrecido por la matriz de precedencias. Tomando en cuenta que si existen aristas requeridos en el camino entre el último nodo de la

solución parcial y el depósito, esta será atendida siempre que cumpla con la restricción de capacidad del vehículo. La Figura 3.10 muestra una representación gráfica de una solución factible después de la etapa de construcción.

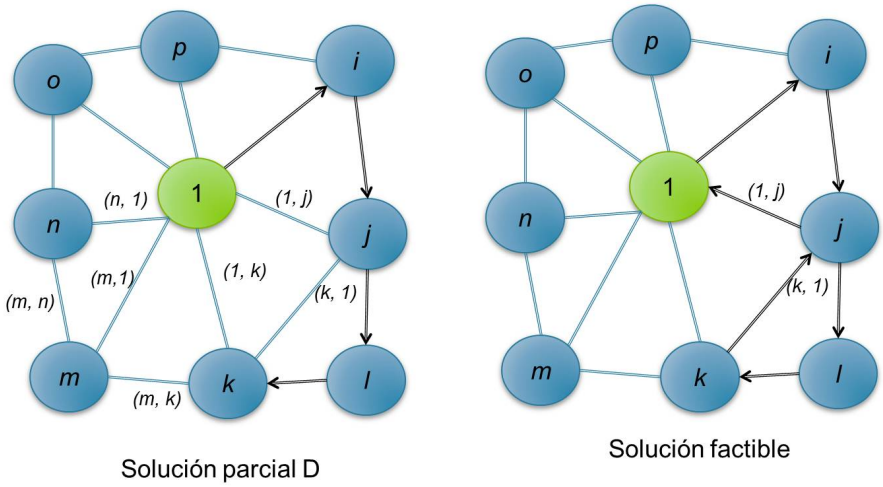


Figura 3.10: Después de haber construido la solución parcial D, no existe ninguna arista requerida que pueda ser atendida por el vehículo por tal motivo el nodo k deberá ser conectado al depósito, se evalúa nuevamente cual es el camino con menor costo de traslado desde el nodo k hacia el nodo *1 (depósito)*, resultando el mejor el camino $k-->l-->1$, por consiguiente una solución factible es como se muestra $1-->i-->j-->l-->k-->j-->1$.

Cada vez que la demanda de una arista requerida sea atendida, esta se elimina de la LRC. Cuando no se puedan agregar mas aristas requeridas al vehículo (por no cumplir con la capacidad de carga), esta ruta se conecta del depósito y se empieza con una nueva ruta. Esta etapa se termina hasta que la LRC este vacía, es decir, se atendió a la totalidad de las aristas requeridas.

La Figura 3.11 muestra una solución factible proporcionada por el programa en la etapa de construcción de una instancia de 27 vértices, 46 aristas requeridas, 10 vehículos disponibles con una capacidad de carga de 27 cada uno y una demanda total requerida de 249. Se puede observar que todas las rutas inician y terminan en el depósito como lo indica el CARP.

```
1--(4)--> 27--(12)--> 26--(13)--> 25--(17)--> 21--(19)--> 22--(25)--> 23--(27)--> 26--(27)--> 27--(27)--> 1  costo=44.00
1--(2)--> 2--(11)--> 12--(13)--> 13--(14)--> 14--(19)--> 20--(21)--> 15--(26)--> 1  costo=33.00
1--(1)--> 24--(10)--> 23--(16)--> 21--(21)--> 14--(24)--> 16--(25)--> 17--(25)--> 1  costo=46.00
1--(8)--> 3--(16)--> 8--(21)--> 9--(24)--> 5--(24)--> 6--(27)--> 3--(27)--> 1  costo=35.00
1--(8)--> 17--(15)--> 15--(22)--> 13--(22)--> 12--(22)--> 2--(22)--> 1  costo=20.00
1--(0)--> 3--(0)--> 6--(1)--> 7--(8)--> 11--(13)--> 10--(21)--> 6--(25)--> 8--(25)--> 3--(25)--> 1  costo=46.00
1--(0)--> 3--(0)--> 6--(9)--> 5--(18)--> 4--(27)--> 6--(27)--> 3--(27)--> 1  costo=33.00
1--(0)--> 2--(0)--> 12--(0)--> 13--(0)--> 14--(9)--> 18--(18)--> 21--(26)--> 20--(26)--> 15--(26)--> 1  costo=33.00
1--(0)--> 3--(0)--> 6--(9)--> 9--(9)--> 5--(9)--> 4--(13)--> 10--(13)--> 4--(17)--> 3--(17)--> 1--(24)--> 19--(24)--> 1  costo=40.00
1--(0)--> 24--(9)--> 27--(9)--> 26--(9)--> 25--(16)--> 18--(16)--> 25--(20)--> 22--(20)--> 23--(20)--> 24--(20)--> 1  costo=39.00
```

Figura 3.11: Ejemplo de solución factible en la etapa de construcción generada por el programa de la instancia de prueba: gdb8.

3.2.3. Etapa de mejora del GRASP aplicada al CARP

En esta etapa se busca mejorar la solución generada por la etapa de anterior mediante la acción de reducción del costo total del recorrido, esto se hace aplicando tres operadores de búsqueda local de la siguiente forma y en el siguiente orden:

1. **Operador unión de rutas con mayor costo**: consiste en seleccionar las 2 rutas con mayor costo de traslado generadas por una solución factible, de ambas rutas son seleccionadas solo los arcos atendidos (es decir aquellos que fue cubierta su demanda) ya sea en una ruta o en la otra como se ilustra en la Figura 3.12.

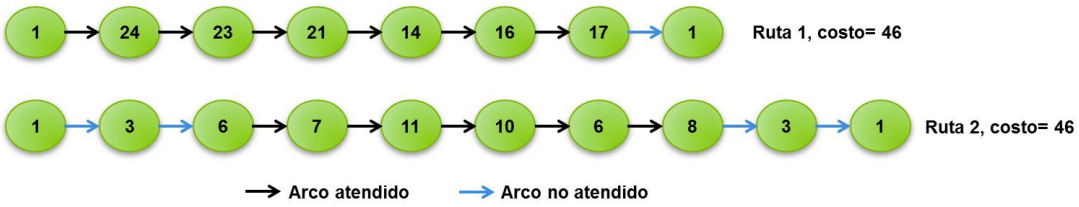


Figura 3.12: La ruta 1 y ruta 2 fueron seleccionadas de la solución factible generada en la etapa de construcción mostrada en la Figura 3.11 y corresponden a las dos rutas mas costosas, ambas con un costo de 46. En base a estas rutas el programa selecciona solo los arcos atendidos de cada ruta y genera una lista de aristas requeridas, en este caso son: $(1, 24), (24, 23), (23, 21), (21, 14), (14, 16), (16, 17), (6, 7), (7, 11), (11, 10), (10, 6)$ y $(6, 8)$ cada una con su respectiva demanda y costo de recorrerla.

   Posteriormente el programa convierte este conjunto de aristas en un subproblema y aplica nuevamente la etapa de construcción. Si encuentra un par de rutas con el costo de ambas menor a la suma de los costos de las dos rutas anteriores, estas son sustituidas por las nuevas rutas y es generada una nueva solución factible. Esto se repite mientras el costo total de la solución factible es mejorado.

2. **Operador unión de rutas con menor costo**: Este algoritmo sigue los mismos pasos que algoritmo unión de rutas con menor costo con la diferencia que en este, las 2 rutas a seleccionar son la que tengan los menores costos de traslado, un ejemplo al respecto se muestra en la Figura 3.13.

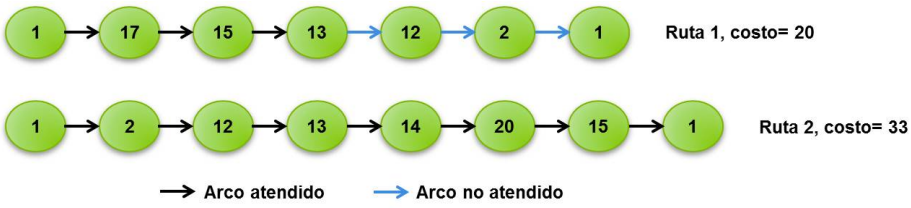


Figura 3.13: La ruta 1 y ruta 2 corresponden a las dos rutas con menor costo, 20 y 33 respectivamente. Con base a estas rutas el programa selecciona solo los arcos atendidos de cada una y genera una lista de aristas requeridas, para este ejemplo son: $(1, 17), (17, 15), (15, 13), (1, 2), (2, 12), (12, 13), (13, 14), (14, 20), (20, 15)$ y $(15, 1)$ cada una con su respectiva demanda y costo de recorrerla.

3. **Operador combinaciones de dos rutas**: se basa en seleccionar combinaciones de 2 rutas (sin importar su costo) de la mejor solución (menor costo) generada en esta etapa del proceso, es decir la ruta 1 combinada con la 2, 3, 4, ...,n (donde n es el total de rutas en la solución), la ruta 2 combinada con la 3, 4, 5, ..., n, y así sucesivamente, en cada combinación son seleccionadas solo los arcos atendidos, en la Figura 3.14 se explica un ejemplo.
   Si en cualquier combinación encuentra un costo menor al anterior este es almacenado y se procede a realizar la siguiente combinación guardando cada vez la mejor reducción.
   Esto se repite hasta que se hayan realizado todas las combinaciones, al final el programa regresa el par de rutas con la mejor reducción la cual sustituirá a las anteriores y una nueva solución es generada (la de menor costo hasta el momento).

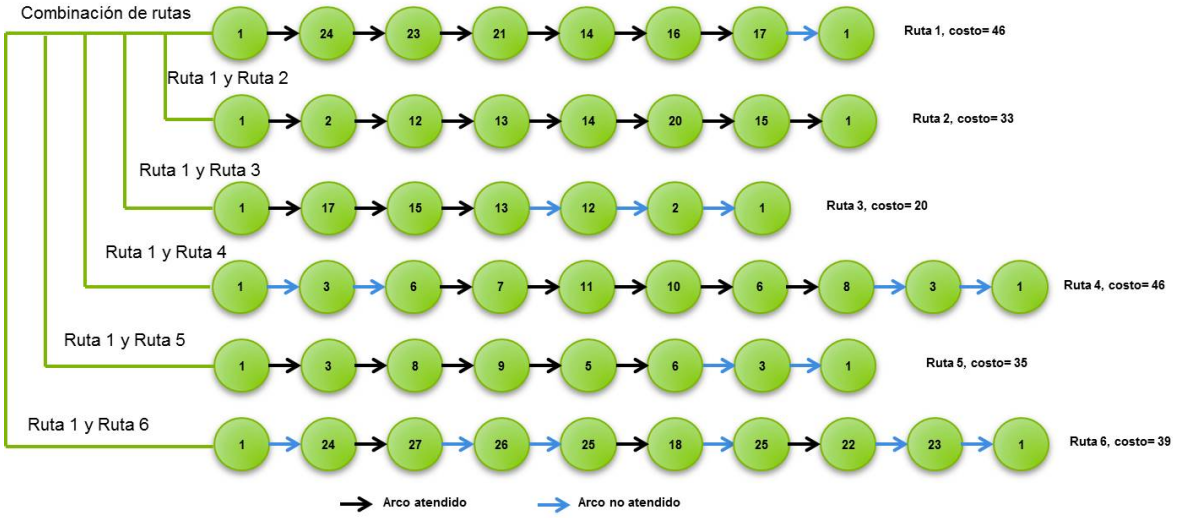


Figura 3.14: En esta solución de 6 rutas, se muestra el ejemplo donde la ruta 1 es unida con la ruta 2, después con la ruta 3, luego con la ruta 4, posteriormente con la ruta 5 y finalmente con la ruta 6, en cada combinación el programa selecciona solo los arcos atendidos de cada una y genera una lista de aristas requeridas para después aplicar la etapa de construcción.

Capítulo 4

Pruebas y resultados

El algoritmo se programo en C++ y fue ejecutado en una laptop con procesador Intel, Core i7-4710HQ, 2.50 HHz, 8 Gb de Ram y con un sistema operativo windows 8. Para evaluar el algoritmo GRASP se utilizaron diferentes instancias de prueba con diferentes tamaños de aristas requeridas propuestas por M. Kiuchi (KSHS) [10], B.L. Golden (GDB) [5] y E. Benavent (VAL) [11] tal como se muestra en la Tabla 4.1.

Talbi (2009) [72], menciona que se deben realizar varias pruebas (al menos 10, más de 100 si es posible) con el fin de obtener resultados estadísticos significativos, por esta razón cada instancia evaluada se prueba 10 veces en el algoritmo, para cada prueba se calcula el % GAP (calidad de la solución), que se refiere la brecha entre la solución obtenida por el algoritmo y la solución óptima o mejor conocida de la instancia de prueba y se calcula de la siguiente manera:

$$\%GAP = \frac{\text{Solución del algoritmo} - \text{Solución mejor conocida}}{\text{Solución mejor conocida}} * 100 \qquad (4.1)$$

Con el fin de determinar los valores iniciales para el alfa, se probo el algoritmo en un 20 % del total de las instancias (seleccionadas aleatoriamente) valores de $\alpha = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6.0.7, 0.8, 0.9, 1.0$ en la etapa de construcción con 25,000 iteraciones (condición de parada), observándose que los valores más bajos en el $\%GAP$ se encuentran en los α de 0.1, 0.2 y 0.3, como se observa en la Figura 4.1.

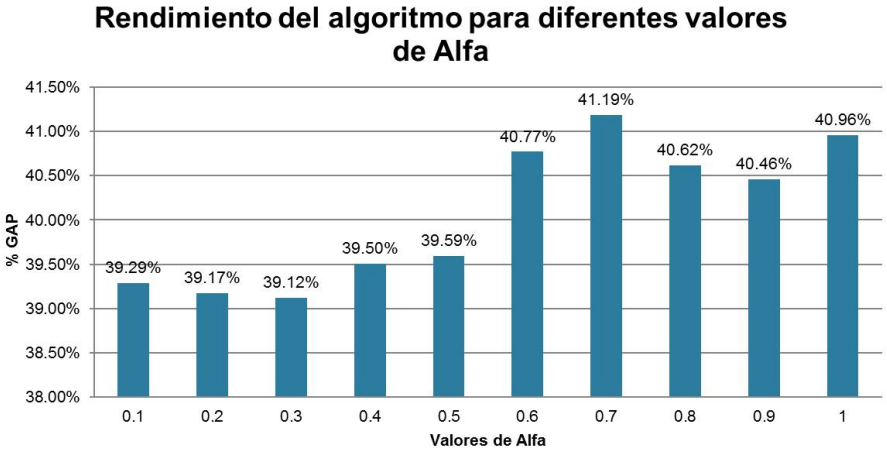


Figura 4.1: Gráfica de valores de α vs % GAP en la etapa de construcción.

Posteriormente se realizaron corridas piloto en la etapa de construcción del GRASP con los siguientes parámetros:

- Condición de parada 25,000 iteraciones.
- Valores de α de 0.1, 0.3 y combinados (0.3, 0.1 y 0.2).
- Dos formas de ordenar las aristas requeridas en base a los costos de traslado de una de menor a mayor y otra de mayor a menor.

Instancias de prueba													
No.	Instancia	$\|V\|$	$\|E\|$	K	Q	Mejor conocido	No.	Instancia	$\|V\|$	$\|E\|$	K	Q	Mejor conocido
1	Kshs1	8	15	4	150	14661	33	Val2A	24	34	2	180	227
2	Kshs2	10	15	4	150	9863	34	Val2B	24	34	3	120	259
3	Kshs3	6	15	4	150	9320	35	Val2C	24	34	8	40	457
4	Kshs4	8	15	4	150	11498	36	Val3A	24	35	2	80	81
5	Kshs5	8	15	3	150	10957	37	Val3B	24	35	3	50	87
6	Kshs6	9	15	3	150	10197	38	Val3C	24	35	7	20	138
7	Gdb1	12	22	5	5	316	39	Val4A	41	69	3	225	400
8	Gdb2	12	26	6	5	339	40	Val4B	41	69	4	170	412
9	Gdb3	12	22	5	5	275	41	Val4C	41	69	5	130	428
10	Gdb4	11	19	4	5	287	42	Val4D	41	69	9	75	528
11	Gdb5	13	26	6	5	377	43	Val5A	34	65	3	220	423
12	Gdb6	12	22	5	5	298	44	Val5B	34	65	4	165	446
13	Gdb7	12	22	5	5	325	45	Val5C	34	65	5	130	474
14	Gdb8	27	46	10	27	348	46	Val5D	34	65	9	75	575
15	Gdb9	27	51	10	27	303	47	Val6A	31	50	3	170	223
16	Gdb10	12	25	4	10	275	48	Val6B	31	50	4	120	233
17	Gdb11	22	45	5	50	395	49	Val6C	31	50	10	50	317
18	Gdb12	13	23	7	35	458	50	Val7A	40	66	3	200	279
19	Gdb13	10	28	6	41	536	51	Val7B	40	66	4	150	283
20	Gdb14	7	21	5	21	100	52	Val7C	40	66	9	65	334
21	Gdb15	7	21	4	37	58	53	Val8A	30	63	3	200	386
22	Gdb16	8	28	5	24	127	54	Val8B	30	63	4	150	395
23	Gdb17	8	28	5	41	91	55	Val8C	30	63	9	65	521
24	Gdb18	9	36	5	37	164	56	Val9A	50	92	3	235	323
25	Gdb19	8	11	3	27	55	57	Val9B	50	92	4	175	326
26	Gdb20	11	22	4	27	121	58	Val9C	50	92	5	140	332
27	Gdb21	11	33	6	27	156	59	Val9D	50	97	10	250	387
28	Gdb22	11	44	8	27	200	60	Val10A	50	97	3	250	428
29	Gdb23	11	55	10	27	233	61	Val10B	50	97	4	190	436
30	Val1A	24	39	2	200	173	62	Val10C	50	97	5	150	446
31	Val1B	24	39	3	120	173	63	Val10D	50	97	10	75	525
32	Val1C	24	39	8	45	245							

Tabla 4.1: Instancias de prueba utilizadas, para cada una se muestra el nombre, número de vértices, cantidad de aristas requeridas, número de vehículos, la capacidad de carga de los vehículos y el mejor resultado conocido el cual se refiere al menor costo total del recorrido.

4.1. Resultados etapa construcción

Para cada grupo de instancias se muestran los resultados de 6 combinaciones en los parámetros del α y el ordenamiento de las aristas requeridas. En la columna A se uti-

lizo un $\alpha = 0.1$ con ordenamiento de menor a mayor costo de traslado, la columna B el $\alpha = 0.1$ pero con ordenamiento de mayor a menor costo de transporte, la columna C corresponde a un $\alpha = 0.3$ con ordenamiento de menor a mayor, la columna D se refiere a $\alpha = 0.3$ pero con ordenamiento de mayor a menor, la columna E se corrió con α combinado = (0.3, 0.1, 0.2) con ordenamiento de menor a mayor y la columna F con α combinado = (0.3, 0.1, 0.2) con ordenamiento de mayor a menor.

En la Tabla 4.2 se muestran los resultados obtenidos para el grupo de instancias Kshs, donde se observa que los mejores valores (menores) del $\%GAP$ están en la columna F con un promedio de 3.09 %, en resumen los valores del $\%GAP$ para cada una de las columnas son graficados en la Figura 4.2.

Instancia	α=0.1 Distancia ordenada de menor a mayor A	α=0.1 Distancia ordenada de mayor a menor B	α=0.3 Distancia ordenada de menor a mayor C	α=0.3 Distancia ordenada de mayor a menor D	α=0.3, 0.1,0.2 Distancia ordenada de menor a mayor E	α=0.3, 0.1,0.2 Distancia ordenada de mayor a menor F
kshs1	17.33 %	13.43 %	11.44 %	5.68 %	3.18 %	1.11 %
kshs2	14.49 %	13.77 %	13.66 %	9.74 %	6.11 %	3.54 %
kshs3	7.77 %	7.39 %	9.56 %	9.08 %	7.35 %	4.63 %
kshs4	8.11 %	7.70 %	8.35 %	7.94 %	5.14 %	5.89 %
kshs5	3.33 %	3.16 %	2.12 %	2.01 %	3.54 %	3.37 %
kshs6	5.19 %	4.13 %	4.29 %	1.54 %	0.00 %	0.00 %
Promedio %GAP=	9.37 %	8.26 %	8.24 %	5.99 %	4.22 %	3.09 %
Máximo %GAP=	17.33 %	13.77 %	13.66 %	9.74 %	7.35 %	5.89 %

Tabla 4.2: Los valores más bajos en el promedio del $\%GAP$ se muestran en la columna F, la cual corresponde a un α combinado = (0.3, 0.1, 0.2) con ordenamiento de mayor a menor costo de traslado, en esta misma columna se aprecia que en una de las instancias es encontrado el óptimo aun sin haber aplicado la etapa de mejora, finalmente se observa que el valor máximo del $\%GAP$ es de 5.89 %.

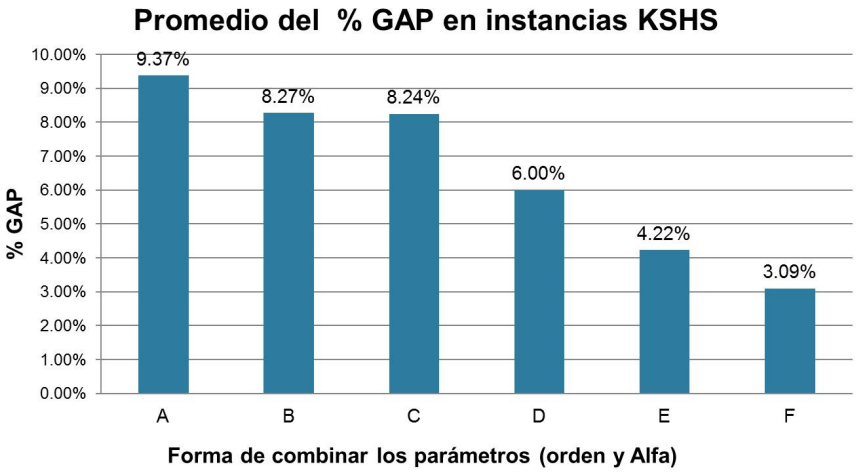


Figura 4.2: Gráfica de promedios del % GAP obtenidos en la etapa de construcción para cada una de las columnas del grupo de instancias KSHS.

De igual manera en la Tabla 4.3 se aprecia que los mejores valores para el % GAP para el grupo de instancias Gdb se encuentran en la columna F con un promedio de 2.48 %. Una mejor apreciación al respecto se muestra en la Figura 4.3 donde se grafican los promedios del %GAP para cada una de las columnas.

Instancia	α=0.1 Distancia ordenada de menor a mayor A	α=0.1 Distancia ordenada de mayor a menor B	α=0.3 Distancia ordenada de menor a mayor C	α=0.3 Distancia ordenada de mayor a menor D	α=0.3, 0.1,0.2 Distancia ordenada de menor a mayor E	α=0.3, 0.1,0.2 Distancia ordenada de mayor a menor F
Gdb1	2.22 %	2.10 %	1.20 %	1.14 %	1.15 %	0.00 %
Gdb2	8.44 %	7.01 %	6.61 %	5.10 %	2.95 %	1.77 %
Gbd3	7.85 %	7.46 %	7.20 %	4.41 %	0.00 %	0.00 %
Gbd4	12.54 %	8.93 %	0.00 %	0.00 %	0.00 %	0.00 %
Gbd5	9.28 %	8.82 %	9.02 %	6.30 %	4.24 %	1.59 %
Gdb6	6.71 %	5.38 %	5.70 %	3.10 %	4.70 %	0.00 %
Gdb7	1.91 %	1.14 %	0.06 %	0.00 %	0.00 %	0.00 %
Gdb8	27.47 %	22.34 %	21.84 %	15.30 %	9.20 %	4.31 %
Gdb9	31.68 %	26.31 %	25.64 %	18.80 %	27.72 %	10.89 %
Gdb10	13.02 %	12.37 %	9.75 %	4.30 %	0.00 %	0.00 %
Gdb11	19.80 %	17.51 %	19.39 %	11.11 %	9.11 %	6.58 %
Gdb12	7.21 %	6.84 %	5.02 %	3.33 %	3.49 %	3.49 %
Gdb13	10.15 %	9.64 %	8.10 %	6.54 %	13.43 %	8.02 %
Gdb14	8.80 %	5.36 %	3.40 %	3.30 %	0.00 %	0.00 %
Gdb15	7.59 %	4.31 %	3.45 %	2.31 %	0.00 %	0.00 %
Gdb16	5.98 %	6.46 %	4.09 %	3.97 %	4.72 %	0.00 %
Gdb17	7.91 %	8.55 %	3.52 %	1.99 %	2.20 %	0.00 %
Gdb18	9.54 %	9.06 %	8.90 %	6.65 %	9.15 %	4.88 %
Gdb19	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Gdb20	12.07 %	7.46 %	5.12 %	3.19 %	08.30 %	0.00 %
Gdb21	15.64 %	14.86 %	13.46 %	8.74 %	7.05 %	4.49 %
Gdb22	11.30 %	8.54 %	7.80 %	7.41 %	7.00 %	2.50 %
Gdb23	18.11 %	13.55 %	13.99 %	9.15 %	14.16 %	8.58 %
Promedio %GAP=	11.10 %	9.30 %	7.97 %	5.48 %	5.59 %	2.48 %
Máximo %GAP=	31.68 %	26.31 %	25.64 %	18.80 %	27.72 %	10.89 %

Tabla 4.3: Los valores más bajos en el promedio del % GAP se muestran en la columna F, la cual corresponde a un α combinado = (0.3, 0.1, 0.2) con ordenamiento de mayor a menor costo de traslado, en esta misma columna se aprecia que en 12 de las instancias es encontrado el óptimo aun sin haber aplicado la etapa de mejora.

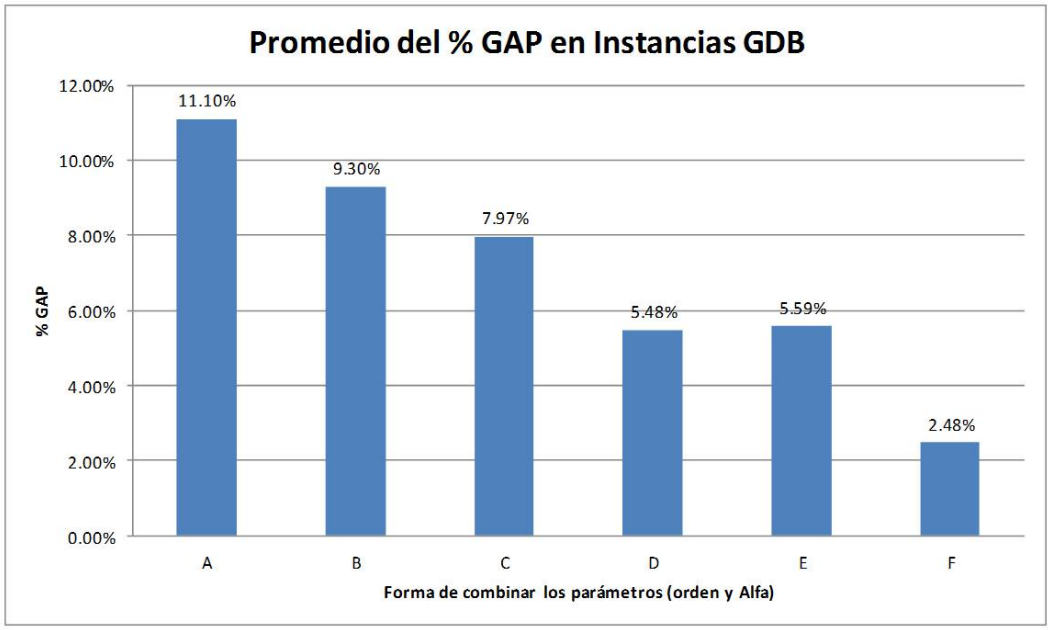


Figura 4.3: Gráfica de promedios del % GAP obtenidos en la etapa de construcción para cada una de las columnas del grupo de instancias GDB.

Al igual que los grupos de instancias Kshs y Gdb, los mejores resultados del % GAP para el grupo de instancias VAL se encontraron en la columna F con un promedio de 49.93 % tal como se observa en la Tabla 4.4.

Instancia	α=0.1 Distancia ordenada de menor a mayor A	α=0.1 Distancia ordenada de mayor a menor B	α=0.3 Distancia ordenada de menor a mayor C	α=0.3 Distancia ordenada de mayor a menor D	α=0.3, 0.1,0.2 Distancia ordenada de menor a mayor E	α=0.3, 0.1,0.2 Distancia ordenada de mayor a menor F
Val1A	42.79 %	35.66 %	34.62 %	31.47 %	28.61 %	26.01 %
Val1B	66.57 %	55.47 %	56.86 %	48.96 %	44.51 %	40.46 %
Val1C	27.08 %	47.56 %	46.18 %	41.98 %	38.16 %	34.69 %
Val2A	62.33 %	51.94 %	50.43 %	45.84 %	41.67 %	37.89 %
Val2B	23.50 %	19.58 %	19.01 %	17.29 %	15.71 %	14.29 %
Val2C	21.55 %	17.96 %	17.44 %	15.85 %	14.41 %	13.10 %
Val3A	52.81 %	44.01 %	42.72 %	38.84 %	35.31 %	32.10 %
Val3B	49.16 %	40.97 %	39.78 %	36.16 %	32.87 %	29.89 %
Val3C	17.99 %	16.35 %	15.83 %	14.39 %	13.08 %	11.89 %
Val4A	114.75 %	95.62 %	92.84 %	84.40 %	76.73 %	69.75 %
Val4B	19.80 %	17.51 %	19.39 %	11.11 %	9.11 %	6.58 %
Val4C	123.00 %	102.50 %	99.51 %	90.47 %	82.24 %	74.77 %
Val4D	25.38 %	25.13 %	24.40 %	22.18 %	20.16 %	18.33 %
Val5A	99.17 %	82.64 %	80.24 %	72.94 %	66.31 %	60.28 %
Val5B	82.99 %	69.16 %	67.15 %	61.04 %	55.49 %	50.45 %
Val5C	106.08 %	88.40 %	85.83 %	78.02 %	70.93 %	64.48 %
Val5D	73.21 %	61.01 %	59.23 %	53.85 %	48.95 %	44.50 %
Val6A	104.76 %	87.30 %	84.75 %	77.05 %	70.04 %	63.68 %
Val6B	83.31 %	69.43 %	67.41 %	61.28 %	55.71 %	50.64 %
Val6C	26.25 %	21.87 %	21.24 %	19.31 %	17.55 %	13.50 %
Val7A	158.03 %	131.69 %	127.85 %	116.23 %	105.66 %	96.06 %
Val7B	151.14 %	125.95 %	122.28 %	111.17 %	101.06 %	91.87 %
Val7C	193.06 %	160.88 %	156.19 %	141.99 %	129.09 %	117.35 %
Val8A	105.27 %	87.73 %	85.17 %	77.43 %	70.39 %	63.99 %
Val8B	69.14 %	57.61 %	55.94 %	50.85 %	46.23 %	42.03 %
Val8C	30.70 %	25.58 %	24.84 %	22.58 %	20.53 %	18.66 %
Val9A	116.13 %	96.77 %	93.95 %	85.41 %	77.65 %	70.59 %
Val9B	129.19 %	107.66 %	104.52 %	95.02 %	86.38 %	78.53 %
val9C	93.58 %	77.99 %	75.71 %	68.83 %	62.57 %	56.89 %
Val9D	19.58 %	16.31 %	15.84 %	14.40 %	13.09 %	11.90 %
Val10A	170.16 %	141.80 %	137.67 %	125.16 %	113.78 %	103.43 %
Val10B	125.89 %	104.91 %	101.85 %	92.59 %	84.18 %	76.52 %
Val10C	88.31 %	73.59 %	71.44 %	64.95 %	59.04 %	53.68 %
Val10D	17.74 %	14.79 %	14.35 %	13.05 %	11.86 %	10.78 %
Promedio %GAP=	82.07 %	68.55 %	66.55 %	60.50 %	60.25 %	49.93 %
Máximo %GAP=	193.06 %	160.88 %	156.19 %	141.99 %	129.09 %	117.35 %

Tabla 4.4: Para el grupo de instancias VAL los valores más bajos en el promedio del %GAP se muestran en la columna F, la cual corresponde a un α combinado = (0.3, 0.1, 0.2) con ordenamiento de mayor a menor costo de traslado.

En este primer análisis para los 3 grupos de instancias, los mejores resultados del %

GAP se encontraron en la columna F que corresponden a un α combinado = (0.3, 0.1, 0.2) con ordenamiento de mayor a menor costo de traslado así como lo muestra en la Figura 4.4. Por lo que se concluye que los parámetros mas convenientes para correr el algoritmo completo (etapa de construcción y etapa de mejora) corresponden a los indicados en la columna F.

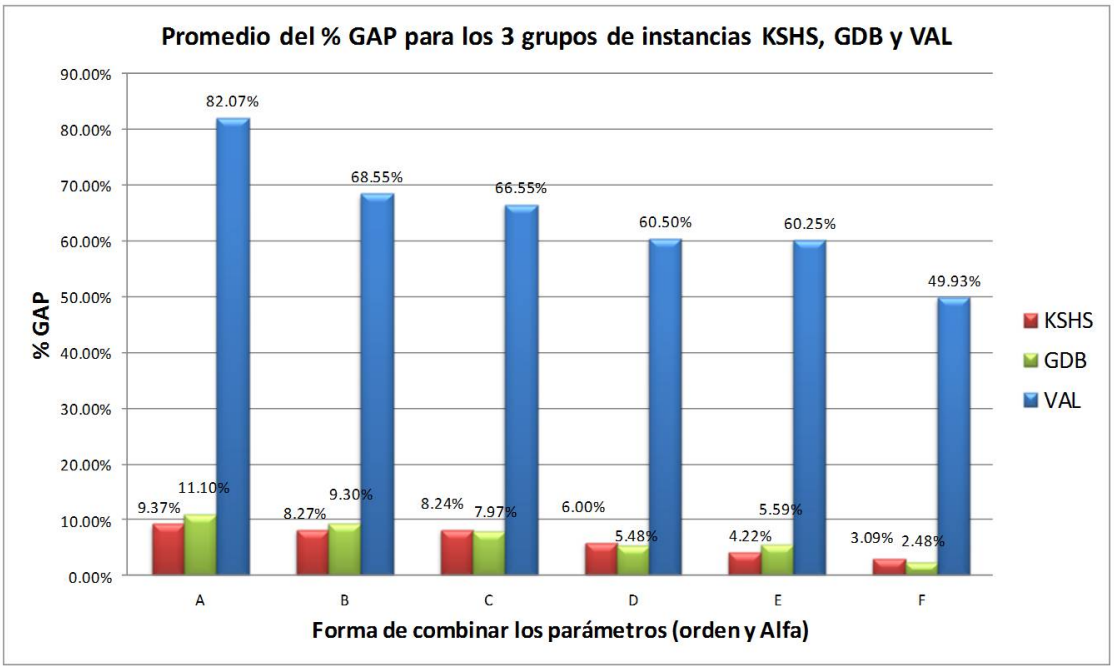


Figura 4.4: En la gráfica se ve que los valores mas bajos en el promedio del % GAP para los 3 grupos de instancias, se muestran en la columna F, la cual corresponde a un α combinado = (0.3, 0.1, 0.2) con ordenamiento de mayor a menor costo de traslado.

4.2. Resultados con búsqueda local

En base al análisis de los resultados obtenidos en la etapa de construcción, se corrió el algoritmo completo con los siguientes parámetros:

- Condición de parada 120,000 iteraciones.
- Valores de α combinados (0.3, 0.1 y 0.2). Esto se refiere a que si la cantidad de aristas aun no atendidas es mayor al 50 % del total de las aristas utilizar un α de 0.3, si la cantidad de aristas que aun no han sido atendidas es menor al 50 % y mayor al 30 % del total de aristas utilizar un α de 0.1 y cuando la cantidad de aristas aun sin atender sea menor a un 30 % del total de aristas utilizar un α de 0.2. Por ejemplo para un problema con 100 aristas requeridas: un α de 0.3 será

utilizado cuando se disponga de 100 a 50 artistas, un α de 0.1 se utiliza cuando queden de 49 a 30 aristas y un α de 0.2 cuando queden 29 o menos aristas.

- Ordenar las aristas requeridas de mayor a menor costo de recorrerla.

Los resultados computacionales se muestran en la columna de %GAP en la Tabla 4.5.

No.	*Instancia*	*|V|*	*|E|*	*Mejor conocido*	*%GAP*	*No.*	*Instancia*	*|V|*	*|E|*	*Mejor conocido*	*%GAP*
1	Kshs1	8	15	14661	0.00 %	33	Val2A	24	34	227	10.25 %
2	Kshs2	10	15	9863	0.00 %	34	Val2B	24	34	259	6.16 %
3	Kshs3	6	15	9320	0.00 %	35	Val2C	24	34	457	5.37 %
4	Kshs4	8	15	11498	3.41 %	36	Val3A	24	35	81	4.64 %
5	Kshs5	8	15	10957	0.00 %	37	Val3B	24	35	87	2.25 %
6	Kshs6	9	15	10197	0.00 %	38	Val3C	24	35	138	0.97 %
7	Gdb1	12	22	316	0.00 %	39	Val4A	41	69	400	35.22 %
8	Gdb2	12	26	339	0.35 %	40	Val4B	41	69	412	24.33 %
9	Gdb3	12	22	275	0.00 %	41	Val4C	41	69	428	26.50 %
10	Gdb4	11	19	287	0.00 %	42	Val4D	41	69	528	13.52 %
11	Gdb5	13	26	377	0.95 %	43	Val5A	34	65	423	21.99 %
12	Gdb6	12	22	298	0.00 %	44	Val5B	34	65	446	14.80 %
13	Gdb7	12	22	325	0.00 %	45	Val5C	34	65	474	15.61 %
14	Gdb8	27	46	348	2.30 %	46	Val5D	34	65	575	8.22 %
15	Gdb9	27	51	303	2.64 %	47	Val6A	31	50	223	23.04 %
16	Gdb10	12	25	275	0.00 %	48	Val6B	31	50	233	12.32 %
17	Gdb11	22	45	395	3.75 %	49	Val6C	31	50	317	4.52 %
18	Gdb12	13	23	458	1.05 %	50	Val7A	40	66	279	41.09 %
19	Gdb13	10	28	536	2.61 %	51	Val7B	40	66	283	35.57 %
20	Gdb14	7	21	100	0.00 %	52	Val7C	40	66	334	0.60 %
21	Gdb15	7	21	58	0.00 %	53	Val8A	30	63	386	22.35 %
22	Gdb16	8	28	127	0.00 %	54	Val8B	30	63	395	20.76 %
23	Gdb17	8	28	91	0.00 %	55	Val8C	30	63	521	6.37 %
24	Gdb18	9	36	164	0.00 %	56	Val9A	50	92	323	41.34 %
25	Gdb19	8	11	55	0.00 %	57	Val9B	50	92	326	34.66 %
26	Gdb20	11	22	121	0.00 %	58	Val9C	50	92	332	43.28 %
27	Gdb21	11	33	156	0.00 %	59	Val9D	50	97	387	10.66 %
28	Gdb22	11	44	200	0.00 %	60	Val10A	50	97	428	54.44 %
29	Gdb23	11	55	233	0.34 %	61	Val10B	50	97	436	41.06 %
30	Val1A	24	39	173	10.40 %	62	Val10C	50	97	446	29.25 %
31	Val1B	24	39	173	9.41 %	63	Val10D	50	97	525	8.20 %
32	Val1C	24	39	245	11.84 %						

Tabla 4.5: El primer grupo de instancias (kSHS), de las 6 instancias en 5 se obtiene el mejor conocido y en la restante se obtiene un GAP menor a 4 %. El segundo grupo de instancias (GDB), de las 23 en 15 instancias se obtiene el mejor conocido y en las 8 restantes se obtiene un GAP menor a 4 %. El tercer grupo de instancias (VAL), de las 34 instancias en ninguno se obtuvo el mejor conocido y solo en 3 se obtiene un GAP menor a 4 %.

La Figura 4.5 muestra una gráfica con el resumen de los resultados obtenidos en la Tabla 4.5, donde se aprecia que de las 63 instancias evaluadas con el algoritmo en 20 de ellas se logra el mejor conocido, en 11 instancias más están por debajo del 4 % GAP, mientras que el resto (32 instancias) están por arriba de un 4 % GAP.

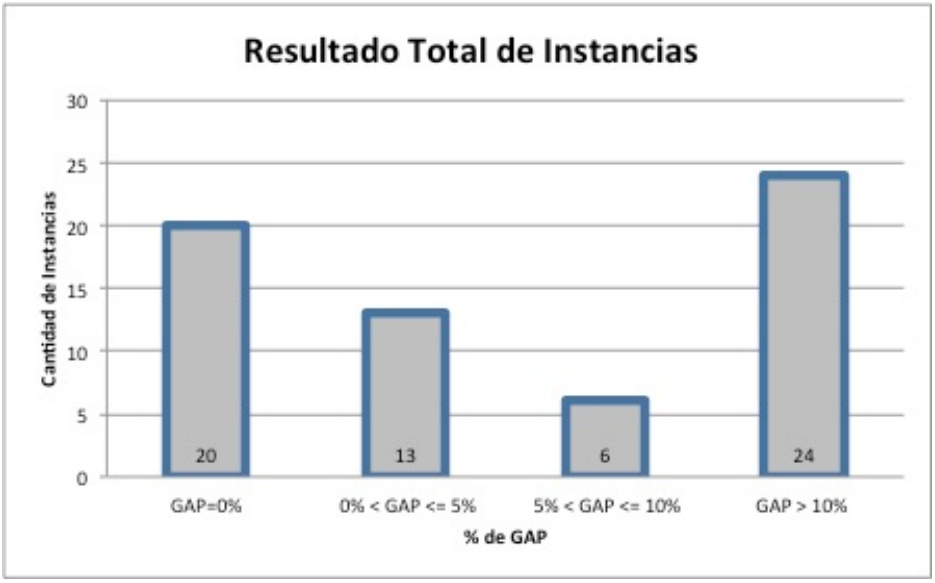


Figura 4.5: La gráfica muestra la cantidad de instancias por intervalo de % GAP, tal como se observa, 33 instancias se encuentran en un intervalo comprendido entre 0 % y 5 % GAP, de las cuales en 20 se encuentra el óptimo, por otro lado, 6 instancias se encuentran entre 5 % y 10 % GAP y 24 de las instancias se encontraron con un % GAP mayor a 10 %.

Finalmente en la Figura 4.6 se muestra la comparación de los resultados con búsqueda local (BL) o sin búsqueda local (BL), aquí se observa que con las búsquedas locales implementadas se reduce el % GAP en los grupos de instancias, de 3.09 % a 0.57 %, de 2.48 % a 0.50 % y de 49.93 % a 19.12 % respectivamente, sin embargo, a medida que aumenta el tamaño del problema, el método de construcción del algoritmo GRASP es limitado, como es el caso de las instancias VAL.

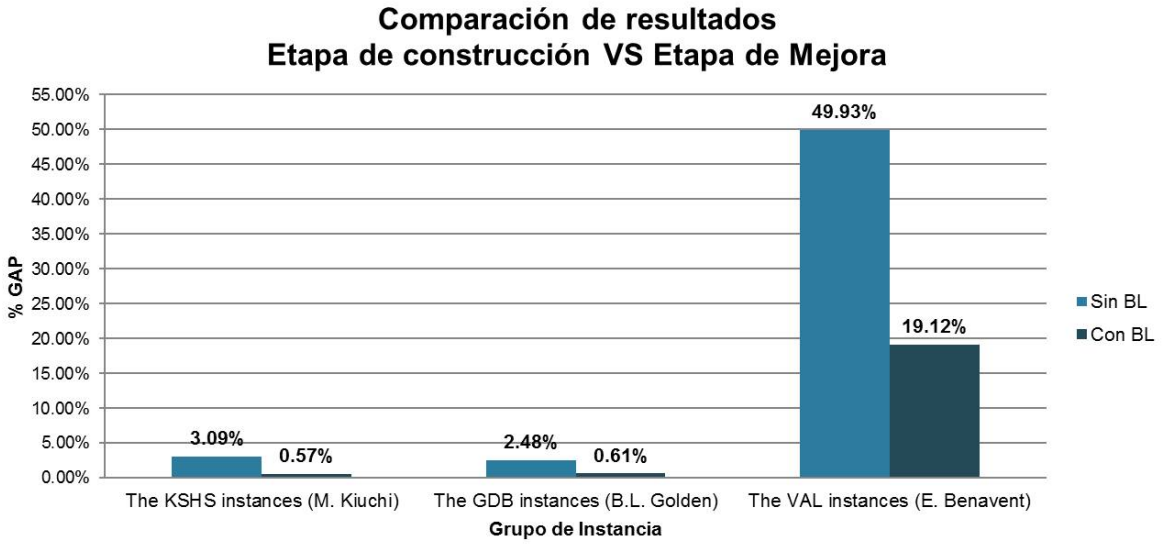


Figura 4.6: Gráfica comparativa de resultados obtenidos con y sin búsqueda local (BL).

4.3. Diseño de experimentos Taguchi

Con el fin de mejorar los resultados presentados hasta el momento se realizó un diseño de experimentos (DOE) con la metodología Taguchi [74] para encontrar una combinación de niveles en los parámetros de entrada (factores) para el algoritmo GRASP, de tal forma que minimice el % del GAP de cada grupo de instancias, principalmente aquellas que ha mostrado un GAP mayor a 10 %.

En la Tabla 4.6 se muestran los factores de control y sus respectivos niveles utilizados en el DOE Taguchi.
El ordenamiento se define en 2 niveles y se refiere a la forma de ordenar la lista de aristas requeridas, la cual puede ser de menor a mayor costo de traslado o mayor a menor costo de traslado. El alfa es considerado en 3 niveles, la cual es utilizada para calcular la cantidad de elementos que contendrá la LRC, en el caso del nivel tres, es cosiderado un alfa mixto. Las iteraciones sin cambio, es definido en 3 niveles, se refiere a que si una solución factible dada no es mejorada después de una cantidad de iteraciones el programa para y regresa la mejor solución encontrada

Factor	*Nivel 1*	*Nivel 2*	*Nivel 3*
Ordenamiento	De menor a mayor costo	De mayor a menor costo	***
Alfa	0.1	0.3	(0.3, 0.1, 0.2)
Iteraciones sin cambio	5500	10500	15500

Tabla 4.6: Factores a considerar en el DOE Taguchi, para cada factor se muestran sus respectivos niveles.

Para este tipo de experimentos de 3 factores con 2 y 3 niveles, Taguchi propone el arreglo ortogonal [75] que se muestra en la Tabla 4.7, del mismo modo Taguchi propone un estadístico llamado razón señal a ruido [74] que se calcula en cada combinación de factores controlables como se indica en la Figura 4.7. La combinación más robusta de los niveles de los factores controlables es la que maximiza la razón señal a ruido (S/R). Debido a que el problema planteado busca encontrar los % GAP con valores más bajos, el DOE Taguchi es analizado como una señal a ruido del tipo ***entre más pequeño es mejor*** y esta dada por 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 la cantidad de grupo de instancias.

i es la corrida del arreglo Taguchi.

Y_i es la media de los datos de los tres grupos de instancias por cada corrida i.

Tipo de característica	Razón señal/ruido (S/R)
Mientras más pequeña es mejor	$-10 \log\left[\frac{1}{n}\Sigma_{i=1}^{n} Y_i^2\right]$
Mientras más grande es mejor	$-10 \log\left[\frac{1}{n}\Sigma_{i=1}^{n} \frac{1}{Y_i^2}\right]$
Su valor nominal es lo mejor (tipo I)	$10 \log\left(\frac{\bar{Y}^2}{S^2}\right)$
Su valor nominal es lo mejor (tipo II)	$-10 \log (S^2)$
Proporción de defectuosos	$-10 \log\left(\frac{p}{(1-p)}\right)$

Figura 4.7: Razones Señal/Ruido para los diferentes tipos de variables de respuesta.

En la Tabla 4.8 se puede ver el arreglo experimental seleccionado y los valores de los promedios del % GAP de cada grupo de instancias para cada corrida, así como la razón señal a ruido.

Arreglo $L_{18}(2 \text{ x } 3^{7-5})$

Configuración	Número de columna							
	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2
3	1	1	3	3	3	3	3	3
4	1	2	1	1	2	2	3	3
5	1	2	2	2	3	3	1	1
6	1	2	3	3	1	1	2	2
7	1	3	1	2	1	3	2	3
8	1	3	2	3	2	1	3	1
9	1	3	3	1	3	2	1	2
10	2	1	1	2	3	2	2	1
11	2	1	2	1	1	3	3	2
12	2	1	3	2	2	1	1	3
13	2	2	1	2	3	1	3	2
14	2	2	2	3	1	2	1	3
15	2	2	3	1	2	3	2	1
16	2	3	1	3	2	3	1	2
17	2	3	2	1	3	1	2	3
18	2	3	3	2	1	2	3	1
1 factor con dos niveles se asignan la columna 1. Los factores con tres niveles se asignan a las columnas restantes: 2, 3, 4, 5, 6, 7, 8.								

Tabla 4.7: Arreglo propuesto por Taguchi, Fuente: Libro análisis y diseño de experimentos [75]. Para el experimento del presente trabajo solo se ocupan las primeras 3 columnas, la columna 1 corresponde al factor *Orden*, la columna 2 al factor *alfa* y la columna 3 para el factor *iteraciones sin cambio*.

Config.	*A*	*B*	*C*	*KSHS*	*GDB*	*VAL*	*Media*	*S/R*
1	1	1	1	8.46 %	4.05 %	29.0 %	13.84 %	15.091
2	1	1	2	8.00 %	3.97 %	27.50 %	13.16 %	15.549
3	1	1	3	8.10 %	3.68 %	29.54 %	13.77 %	14.986
4	1	2	1	0.00 %	1.35 %	26.27 %	9.21 %	16.371
5	1	2	2	0.00 %	1.08 %	26.27 %	9.12 %	16.375
6	1	2	3	0.00 %	1.06 %	26.97 %	9.11 %	16.375
7	1	3	1	1.56 %	1.44 %	25.96 %	9.65 %	16.456
8	1	3	2	0.00 %	1.26 %	24.52 %	8.59 %	16.969
9	1	3	3	0.00 %	1.10 %	24.51 %	8.54 %	16.976
10	2	1	1	2.48 %	1.61 %	23.51 %	9.20 %	17.278
11	2	1	2	2.60 %	1.26 %	22.66 %	8.84 %	17.596
12	2	1	3	2.48 %	1.17 %	22.32 %	8.66 %	17.732
13	2	2	1	0.00 %	0.40 %	19.17 %	6.52 %	19.117
14	2	2	2	0.00 %	0.40 %	18.96 %	6.45 %	19.213
15	2	2	3	0.00 %	0.28 %	17.47 %	5.91 %	19.934
16	2	3	1	0.57 %	0.50 %	19.12 %	6.73 %	19.135
17	2	3	2	0.57 %	0.50 %	18.53 %	6.53 %	19.406
18	2	3	3	0.57 %	0.50 %	18.09 %	6.39 %	19.615

Tabla 4.8: Arreglo L_{18}, para cada grupo de instancias se muestra el promedio de % GAP. La columna A se refiere al factor *orden*, la columna B al factor *alfa* y la columna C al factor *iteraciones sin cambio*.

Con los resultados mostrados en la Tabla 4.8, se comprobó si estos se ajustaban o no a una distribución normal, con un nivel de significancia del 95 %, mediante las pruebas de normalidad de Anderson-Darling (1952) [76]. Las hipótesis para esta prueba son:

- H_0: Los datos siguen una distribución normal.
- H_1: Los datos no siguen una distribución normal.

Donde, H_0 y H_1 son las hipótesis nula y alternativa respectivamente. Esta prueba nos dice que si el valor p es inferior al nivel de significancia de 0.05, se concluye que los datos no siguen una distribución normal.
Los resultados de la prueba se muestra en la Figura 4.8, donde se observa que el valor de P=0.087 es mayor que el nivel de significancia de 0.05, por lo tanto se acepta H_0, es decir, los datos siguen una distribución normal.

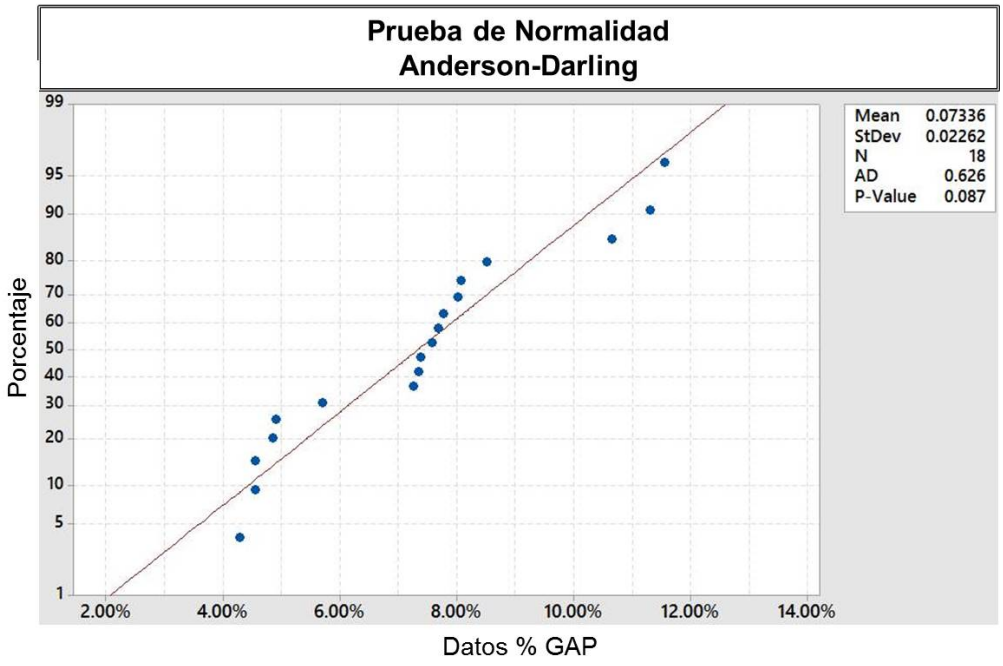


Figura 4.8: Prueba de normalidad Anderson-Darling.

Una vez comprobado que los datos siguen una distribución normal, se procede a analizarlos por medio de las gráficas factoriales para la razón señal a ruido y los promedios, respectivamente, los cuales se elaboraron utilizando el software Minitab versión 15.

La gráfica factorial para la razón señal a ruido se muestran en las Figura 4.9, de aquí se deduce que los efectos orden y α son los que más afectan en la S/R, es decir, el factor orden y α (ambos en su nivel 2) influye bastante sobre la variación del % GAP.

La gráfica factorial para los efectos principales de la variable de respuesta % GAP se muestra en la Figura 4.10, donde se observa que los factores orden en el nivel 2 y α en su nivel 2 son los que tienen más efecto sobre la media de la variable de respuesta % GAP.

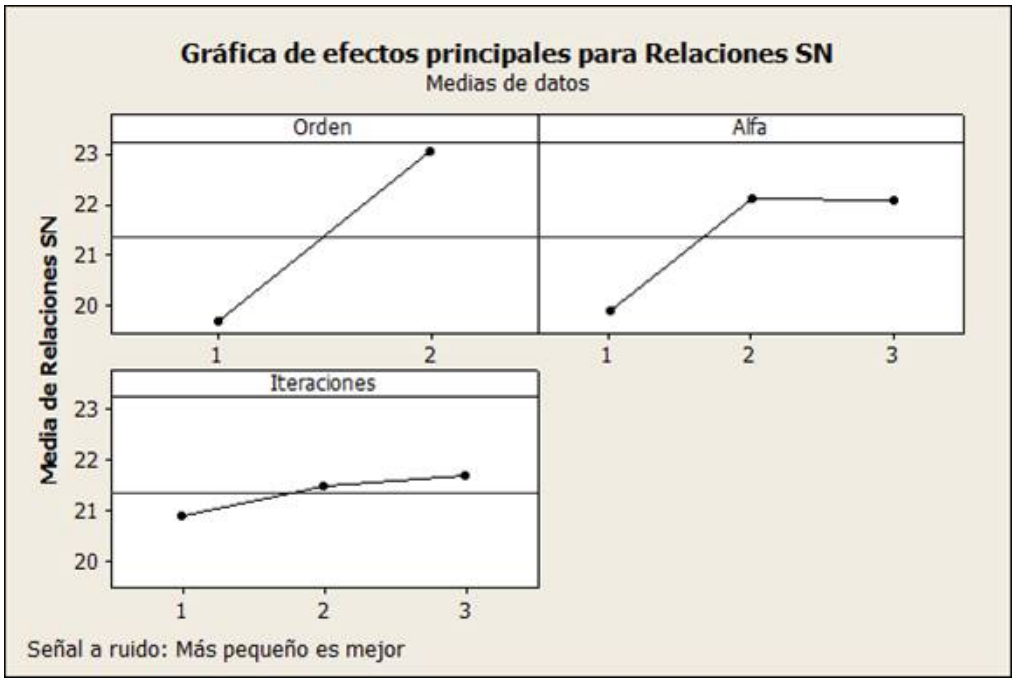


Figura 4.9: Gráfica factorial para la razón señal a ruido.

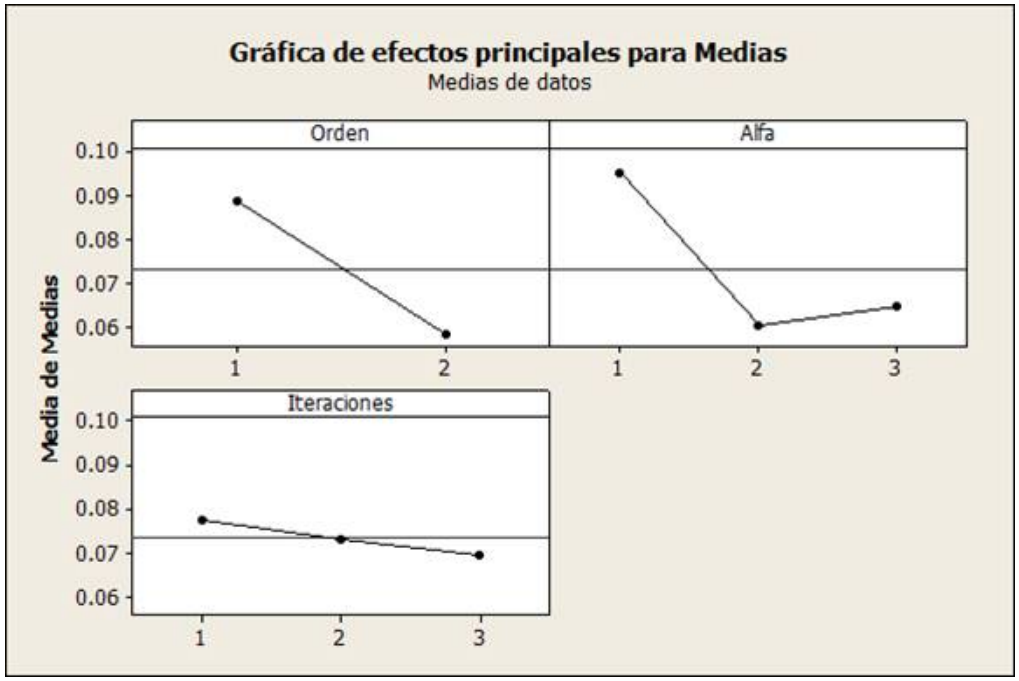


Figura 4.10: Gráfica factorial de efectos principales para medias.

Ademas, en ambas gráficas anteriores, se observa poca pendiente para el factor

iteraciones sin cambio lo que sugiere que el efecto de esta variables no es significativo para ambas respuestas.
Para verificar estadísticamente el efecto de los factores tanto en la razón señal a ruido, como en el % GAP promedio, se realizó el análisis de varianza (ANOVA).
Este análisis es mostrado en las Figuras 4.11 y 4.12 donde se obtuvo como resultado que los factores que afectan significativamente (Valores P menores al nivel de significancia del 5 %) a ambas variables de respuesta son: orden y alfa por lo que es importante que éstos se fijen a sus mejores niveles. Por otro lado, se confirma que el factor *iteraciones sin cambio* resulto ser no significativo.

```
S = 0.2817   R-cuad. = 97.8%   R-cuad.(ajustado) = 96.9%


Análisis de varianza de Relaciones S/R (señal de ruido)

Fuente           GL  SC Sec.  SC Ajust.  MC Ajust.       F      P
Orden             1  31.6761    31.6761    31.6761  399.21  0.000
Alfa              2  10.6511    10.6511     5.3256   67.12  0.000
Iteraciones       2   0.4293     0.4293     0.2147    2.71  0.107
Error residual   12   0.9522     0.9522     0.0793
Total            17  43.7087
```

Figura 4.11: ANOVA de la razón señal a ruido.

```
S = 0.006600   R-cuad. = 95.0%   R-cuad.(ajustado) = 92.9%


Análisis de varianza de Medias

Fuente           GL   SC Sec.  SC Ajust.  MC Ajust.       F      P
Orden             1  0.004917   0.004917   0.004917  112.89  0.000
Alfa              2  0.004941   0.004941   0.002470   56.72  0.000
Iteraciones       2  0.000077   0.000077   0.000038    0.88  0.439
Error residual   12  0.000523   0.000523   0.000044
Total            17  0.010457
```

Figura 4.12: ANOVA de medias.

4.3.1. Resultados del DOE Taguchi

El DOE Taguchi realizado en la sección anterior sugiere que para obtener los % GAP con menores valores se recomienda establecer el nivel de cada parámetro como se muestran en la Tabla 4.9, es decir, Ordenar las aristas requeridas de mayor a menor costo de

transporte con valor de alfa de 0.3 para calcular la cantidad de aristas requeridas que tendrá la LRC y 15500 iteraciones sin cambio como condición de parada.

Tabla 4.9: Niveles sugeridos por DOE Taguchi

Factor	Nivel
Ordenamiento	Mayor a menor
Alfa	0.30
Iteraciones	15500

En la Tabla 4.10 se muestran los resultados obtenidos con los parámetros sugeridos por el DOE Taguchi, aquí se observa que en primer grupo de instancias (kSHS), en todas se obtiene el mejor conocido, es decir un GAP=0.00 %. El segundo grupo (GDB), en todas se logra un GAP menor a 2.30 % de las cuales en 19 instancias se obtiene el mejor conocido. El tercer grupo (VAL), en 15 se obtiene un GAP menor a 10 %, de las cuales en una se encontró el mejor conocido.

La Figura 4.13 muestra una gráfica con el resumen de los resultados obtenidos en la Tabla 4.10, donde se aprecia que de las 63 instancias evaluadas con el algoritmo en 26 de ellas se logra el mejor conocido, 18 instancias están por debajo del 10 % GAP, mientras que el resto (19 instancias) están por arriba de un 10 %GAP.

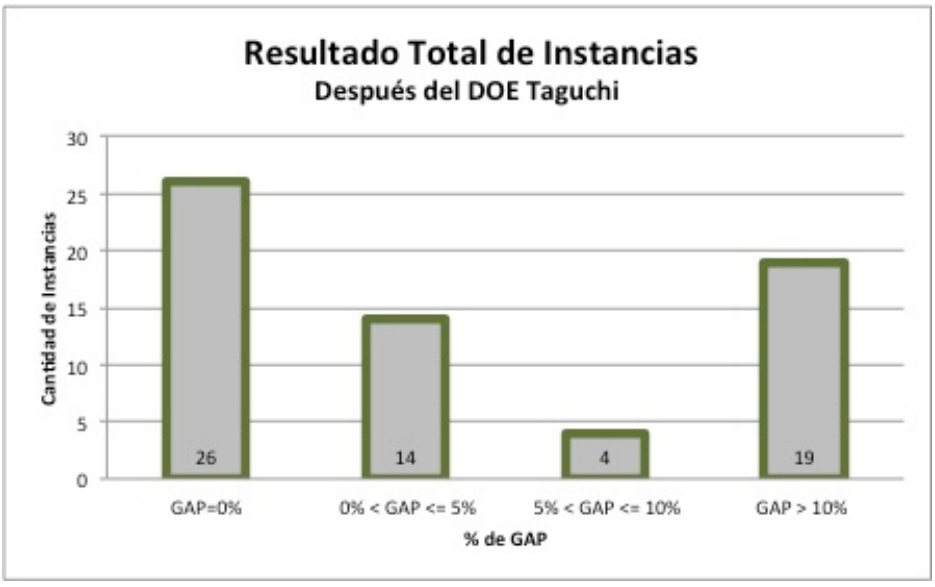


Figura 4.13: Gráfica después del DOE Taguchi, se muestra que 44 instancias se encuentran en un intervalo comprendido entre 0 % y 10 % GAP y las 19 restantes se encontraron con un % GAP mayor a 10 %.

No.	*Instancia*	$\|V\|$	$\|E\|$	*Mejor conocido*	*%GAP*	*No.*	*Instancia*	$\|V\|$	$\|E\|$	*Mejor conocido*	*%GAP*
1	Kshs1	8	15	14661	0.00 %	33	Val2A	24	34	227	9.25 %
2	Kshs2	10	15	9863	0.00 %	34	Val2B	24	34	259	0.77 %
3	Kshs3	6	15	9320	0.00 %	35	Val2C	24	34	457	3.06 %
4	Kshs4	8	15	11498	0.00 %	36	Val3A	24	35	81	4.64 %
5	Kshs5	8	15	10957	0.00 %	37	Val3B	24	35	87	1.25 %
6	Kshs6	9	15	10197	0.00 %	38	Val3C	24	35	138	0.00 %
7	Gdb1	12	22	316	0.00 %	39	Val4A	41	69	400	20.35 %
8	Gdb2	12	26	339	0.00 %	40	Val4B	41	69	412	14.81 %
9	Gdb3	12	22	275	0.00 %	41	Val4C	41	69	428	19.16 %
10	Gdb4	11	19	287	0.00 %	42	Val4D	41	69	528	4.17 %
11	Gdb5	13	26	377	0.00 %	43	Val5A	34	65	423	21.99 %
12	Gdb6	12	22	298	0.00 %	44	Val5B	34	65	446	13.23 %
13	Gdb7	12	22	325	0.00 %	45	Val5C	34	65	474	15.61 %
14	Gdb8	27	46	348	1.32 %	46	Val5D	34	65	575	3.48 %
15	Gdb9	27	51	303	1.98 %	47	Val6A	31	50	223	17.04 %
16	Gdb10	12	25	275	0.00 %	48	Val6B	31	50	233	2.58 %
17	Gdb11	22	45	395	2.28 %	49	Val6C	31	50	317	1.26 %
18	Gdb12	13	23	458	0.00 %	50	Val7A	40	66	279	43.09 %
19	Gdb13	10	28	536	1.12 %	51	Val7B	40	66	283	30.04 %
20	Gdb14	7	21	100	0.00 %	52	Val7C	40	66	334	0.48 %
21	Gdb15	7	21	58	0.00 %	53	Val8A	30	63	386	22.28 %
22	Gdb16	8	28	127	0.00 %	54	Val8B	30	63	395	20.76 %
23	Gdb17	8	28	91	0.00 %	55	Val8C	30	63	521	4.61 %
24	Gdb18	9	36	164	0.00 %	56	Val9A	50	92	323	41.34 %
25	Gdb19	8	11	55	0.00 %	57	Val9B	50	92	326	39.01 %
26	Gdb20	11	22	121	0.00 %	58	Val9C	50	92	332	44.28 %
27	Gdb21	11	33	156	0.00 %	59	Val9D	50	97	387	5.66 %
28	Gdb22	11	44	200	0.00 %	60	Val10A	50	97	428	47.43 %
29	Gdb23	11	55	233	0.00 %	61	Val10B	50	97	436	40.37 %
30	Val1A	24	39	173	10.40 %	62	Val10C	50	97	446	28.25 %
31	Val1B	24	39	173	7.51 %	63	Val10D	50	97	525	5.71 %
32	Val1C	24	39	245	11.84 %						

Tabla 4.10: Resultados después del DOE Taguchi. De las 63 instancias evaluadas, en 44 se obtiene un GAP menor a 10 %, de las cuales en 26 se encontró el mejor conocido. Las 19 instancias restantes se encuentran por arriba del 10 %GAP

En la gráfica de la Figura 4.14 se aprecia la mejora después de haber aplicado los parámetros sugerios por el DOE Taguchi. Antes del DOE en 20 instancias se encontró el mejor conocido mientras que después del DOE se incremento a 26, de igual manera para las instancias con un % GAP mayor al 10 %, se observa una reducción de 24 a 19 después del DOE.

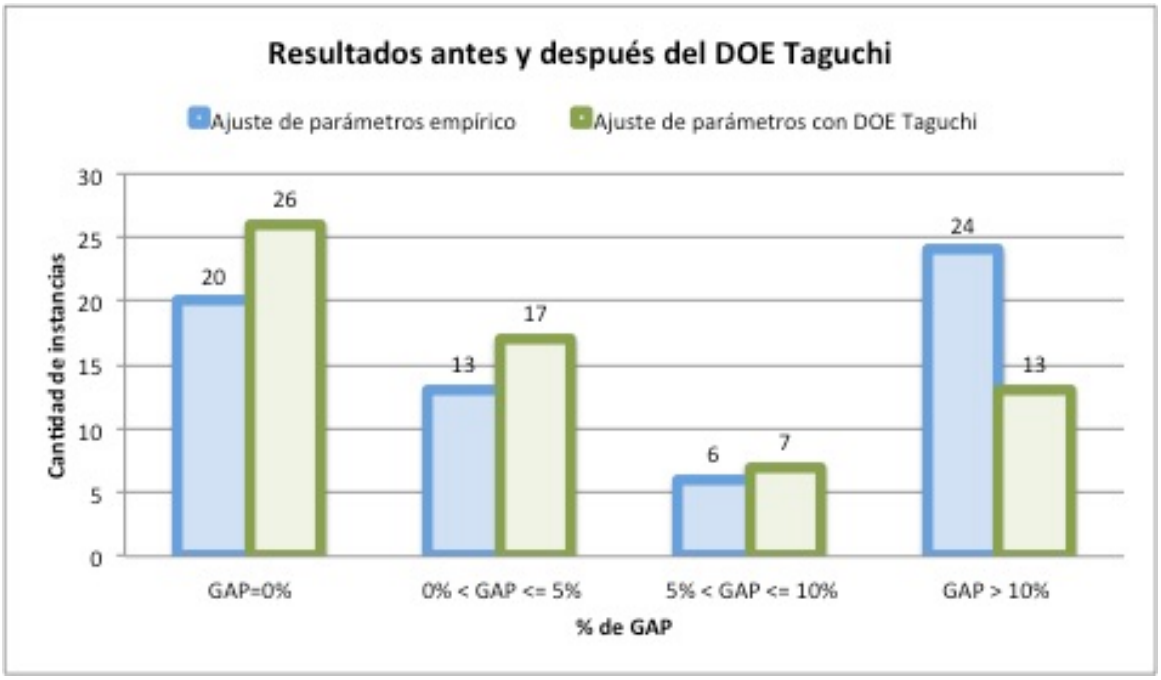


Figura 4.14: Gráfica del antes y después del DOE Taguchi, se observa un incremento en la cantidad de instancias con un GAP menor a 10.0 % GAP de 39 a 44 después de haber aplicado el DOE.

En general, en el análisis de los resultados mostrados después de ajustar los parámetros en los valores propuestos por el DOE Taguchi, se logra una reducción en el % del GAP de los tres grupos de instancias, tal como se muestra en la Tabla 4.11, el grupo KSHS se reduce un 0.57 unidades, además se logra el mejor conocido por la literatura en cada una de las instancias que comprende dicho grupo, en el grupo GBD se logra una reducción 0.32 unidades y en las instancias del grupo VAL la reducción es del 5.63 unidades, por último, de las 63 instancias evaluadas se reduce un 2.17 unidades.

Tabla 4.11: Comparación del promedio del % GAP por grupo de instancias

Grupo	Antes de Taguchi	Después de Taguchi	Reducción
KSHS	0.57 %	0.00 %	0.57
GDB	0.61 %	0.29 %	0.32
VAL	21.97 %	16.34 %	5.63
Total instancias	7.72 %	5.55 %	2.17

4.4. Resultados finales

Finalmente, en base a los resultados mostrados en la gráfica de efectos principales de las medias del factor *iteraciones sin cambio*, Figura 4.10, se observa que a medida que este aumenta se aprecia una ligera tendencia descendente en la media del % GAP, por lo que se decidió explorar el espacio de soluciones y tratar salir de los óptimos locales proporcionados por el algortimo, se incremento el factor *iteraciones sin cambio* y se realizo una última prueba con los factores en los niveles propuestos por el DOE Taguchi pero con 46,500 *iteraciones sin cambio*, es decir tres veces más que el utilizado anteriormente como condición de parada.

Los resultados al aumentar las *iteraciones sin cambio* se muestran en la Tabla 4.12, en donde se observa que en el grupo de instancias (kSHS), en todas se obtiene el mejor conocido, es decir un GAP=0.00 %. En el grupo (GDB), en todas se logra un GAP menor a 1.99 % de las cuales en 19 instancias se obtiene el mejor conocido. En el grupo (VAL), en 21 se obtiene un GAP menor a 10 %, de las cuales en una se encontró el mejor conocido.

No.	*Instancia*	*%GAP*	*No.*	*Instancia*	*%GAP*	*No.*	*Instancia*	*%GAP*
1	Kshs1	0.00 %	22	Gdb16	0.00 %	43	Val5A	16.35 %
2	Kshs2	0.00 %	23	Gdb17	0.00 %	44	Val5B	9.64 %
3	Kshs3	0.00 %	24	Gdb18	0.00 %	45	Val5C	5.95 %
4	Kshs4	0.00 %	25	Gdb19	0.00 %	46	Val5D	4.52 %
5	Kshs5	0.00 %	26	Gdb20	0.00 %	47	Val6A	9.24 %
6	Kshs6	0.00 %	27	Gdb21	0.00 %	48	Val6B	1.55 %
7	Gdb1	0.00 %	28	Gdb22	0.00 %	49	Val6C	1.51 %
8	Gdb2	0.00 %	29	Gdb23	0.00 %	50	Val7A	36.62 %
9	Gdb3	0.00 %	30	Val1A	5.72 %	51	Val7B	20.24 %
10	Gdb4	0.00 %	31	Val1B	4.34 %	52	Val7C	0.30 %
11	Gdb5	0.00 %	32	Val1C	9.02 %	53	Val8A	17.36 %
12	Gdb6	0.00 %	33	Val2A	2.86 %	54	Val8B	13.16 %
13	Gdb7	0.00 %	34	Val2B	0.15 %	55	Val8C	4.57 %
14	Gdb8	1.32 %	35	Val2C	1.36 %	56	Val9A	28.79 %
15	Gdb9	1.98 %	36	Val3A	1.46 %	57	Val9B	25.77 %
16	Gdb10	0.00 %	37	Val3B	1.14 %	58	Val9C	33.01 %
17	Gdb11	0.41 %	38	Val3C	0.00 %	59	Val9D	4.42 %
18	Gdb12	0.00 %	39	Val4A	19.85 %	60	Val10A	42.00 %
19	Gdb13	0.75 %	40	Val4B	11.77 %	61	Val10B	29.31 %
20	Gdb14	0.00 %	41	Val4C	7.38 %	62	Val10C	24.44 %
21	Gdb15	0.00 %	42	Val4D	3.77 %	63	Val10D	5.52 %

Promedio de % GAP por grupo: Kshs=0.00 % , Gdb=0.19 % , Val=11.80 %

Tabla 4.12: Resultados computacionales finales.

Un resumen de los resultados de las Tablas 4.12 y 4.10 es presentado en la gráfica

de la Figura 4.15, donde se aprecia una reducción de 19 a 13 instancias con un GAP mayores a 10 % y un amento de 44 a 50 instancias que se logro un GAP menor a 10 %.

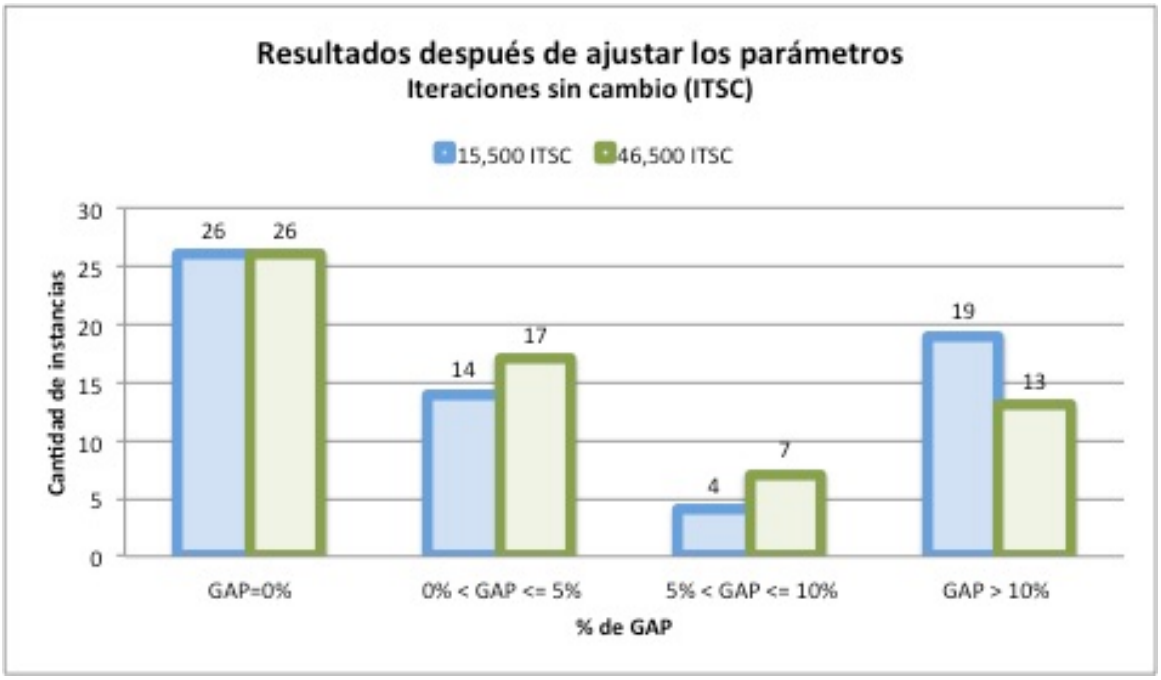


Figura 4.15: Gráfica de resultados con parámetros ajustados con Taguchi.

De igual manera en la en la Tabla 4.13 se observa una reducción en el promedio del % GAP al realizar el ajuste de parámetros mediante un DOE Taguchi comparado con el realizado de forma empírica. En el grupo de instancias KSHS el promedio en el % GAP se logra reducir un 0.57 unidades, en el grupo GBD se logra una reducción 0.42 unidades y en las instancias del grupo VAL la reducción es del 10.17 unidades, en general, de las 63 instancias evaluadas la reducción es de un 3.72 unidades.

Tabla 4.13: Tabla comparativa del promedio del % GAP por grupo de instancias entre el ajuste de parámetros de forma empírico vs ajuste de parámetros mediante DOE Taguchi

Grupo	Ajuste Empírico	Ajuste con DOE Taguchi	Reducción
KSHS	0.57 %	0.00 %	0.57
GDB	0.61 %	0.19 %	0.42
VAL	21.97 %	11.80 %	10.17
Total instancias	7.72 %	4.00 %	3.72

Capítulo 5

Conclusiones generales

5.1. Discusiones

Con los resultados presentados en la Tabla 4.12, se muestra que el algoritmo presenta un alto rendimiento para el grupos de instancias KSHS, donde se logra en cada una de ellas la solución mejor conocida por la literatura. Para el grupo de instancias GDB se obtiene un promedio de 0.19 % GAP, de las cuales en 19 se logra el mejor conocido por la literatura y de las 4 restantes se obtiene un GAP de 1.32 %, 1.98 %, 0.41 % y 0.75 % respectivamente. En el conjunto de instancias VAL, se obtiene un promedio del GAP de 11.80 %, donde, 21 obtienen un GAP menor a 10 % de las cuales en una se logra la solución mejor conocida por la literatura.

Para lograr los promedios más bajos en el % de GAP con la metaheurística propuesta por la presente investigación, se recomienda que los parámetros de entrada para el algoritmo se establezcan como se muestra en la Tabla 5.1, es decir, ordenar las aristas requeridas de mayor a menor costo de transporte, el valor de alfa de 0.3 para calcular la cantidad de aristas requeridas que tendrá la LRC y 46500 iteraciones sin cambio como condición de parada.

Tabla 5.1: Parámetros sugeridos para la metaheurística

Parámetro	Valor recomendado
Ordenamiento	Mayor a menor
Alfa	0.30
Iteraciones	46500

Por otro lado, los resultados obtenidos presentados en la Tabla 4.13 muestran que el desempeño del algoritmo se mejora al establecer los parámetros mediante el ajuste con DOE Taguchi contra un ajuste empírico, tal como se aprecia en las instancias KSHS con ajuste empírico se obtiene un GAP de 0.57 % contra un GAP de 0.00 %

alcanzado por Taguchi, de igual forma para el grupo GDB un GAP de 0.61 % logrado con ajuste empírico contra el obtenido por Taguchi de 0.19 % , lo mismo se observa con las instancias VAL con 21.97 % de GAP con ajuste empírico contra el logrado con Taguchi de 11.80 %.

5.2. Conclusiones

En la presente tesis, se da solución al problema de ruteo por arcos con capacidad limitada en los vehículos mediante una metaheuristica que utiliza la aplicación de un procedimiento de búsqueda basado en funciones voraces, aleatorizadas, adaptativas y se analiza la calidad de las soluciones generadas respecto a las instancias de prueba propuestas por la literatura, ademaś se compara el efecto de ajustar los parámetros con un procedimiento estadístico Taguchi, cumpliendo así con el objetivo general planteado en esta investigación.

Durante el proceso de logar el objetivo general, se cumplen también con los objetivos específicos:

En cuanto al primer objetivo, se realizó una revisión de la literatura, para estudiar y comprender detalladamente el problema del CARP, que abarca desde la definición, orígenes 2.1, modelado matemático 2.2, propuestas de solución (exactos, heurísticos y metaheurísticos) 2.3 y variantes del problema CARP 2.4.

El segundo objetivo especifico, se observa que en los últimos 16 años ha propuestos diferentes metaheurísticas que resuelven el CARP, las cuales se resumen en las siguientes: Tabu Search (TS), Variable Neighborhood Descent (VND), Guide local search (GLS), Ant Colony (AC), Genetic Algorithm (GA), Memetic Algorithm (MA), Greedy Randomized Adaptive Search Procedure e (GRASP) y Biased Random-Key Genetic Algorithm (BRKGA), como se muestran Tabla 2.1, y se detallan en la sección 2.3.3.

Sobre el tercer objetivo especifico, se estudió y analizó el algoritmo metaheurístico GRASP en su forma general como se detalla en la 2.5, posteriormente fué ajustado para generar soluciones al CARP, tal como se explican en la sección 3.2.

Referente al cuarto objetivo especifico, los parámetros de entrada del algoritmo fueron calibrados mediante un diseño de experimentos con la metodología Taguchi con un arreglo ortogonal de 18 configuraciones 4.7.

Finalmente, el quinto objetivo especifico, una vez ajustados los parámetros a los valores proporcionados por el DOE Taguchi, el algoritmo fue ejecutado y comparado en tres grupos de instancias de prueba propuestas por M. Kiuchi (KSHS) [10], B.L. Golden (GDB) [5] y E. Benavent (VAL) [11], en donde se logra la solución mejor conocida por

la literatura en cada una de las instancias KSHS, el en grupo (GDB), se logra un GAP entre un rango de 0.00 % a 1.99 % de las cuales en 19 de 23 se obtiene la solución mejor conocida, para el grupo (VAL), en 21 instancias se obtiene un GAP menor a 10 %, de las cuales en una se encontró la solución mejor conocida.

Con base a los resultados obtenidos por el algoritmo sobre los tres grupos de instancias, se concluye que es un algoritmo competitivo para las instancias de prueba con vértices menores a 30 y aristas requeridas menores a 50, el algoritmo da resultados de buena calidad. En instancias con vértices mayores a 40 y aristas requeridas mayores a 69, el algoritmo da resultados con GAP mayores a 10 %.

Con los parámetros ajustados a los valores propuestos por el DOE Taguchi en vez de ajustados empíricamente, se reduce el promedio del %GAP en los tres grupos de instancias, de 0.57 % a 0.00 %, de 0.61 % a 0.29 % y de 21.97 % a 16.34 % respectivamente, por lo que se recomienda utilizar los parámetros con α de 0.3 y un ordenamiento de mayor menor costo de transporte.

5.3. Trabajo a futuro

Como trabajo futuro se propone realizar el ajuste de parámetros con otros métodos de calibración como ParamILS [77] y Calibra [78], con el fin de buscar valores en los parámetros que mejoren los soluciones para instancias con vértices mayores a 30 y aristas requeridas mayores a 69.

Aplicar el algoritmo propuesto a casos de la vida real, tales como recolección de basura domestica, mantenimiento de calles, lectura de medidores eléctricos, etc., esto a través de convenios de colaboración entre universidad y organismos o empresas de la localidad, con la finalidad de generar transferencias de soluciones tecnológicas, además propiciar los medios para generar nuevos problemas que puedan ser resueltos mediante el ámbito académico.

Por último, incluir en la etapa de mejora del algoritmo propuesto otros operadores de mejora basados en algoritmos genéticos y BRKGA por ejemplo, esto con la intención de generar una exploración más variada en espacio de soluciones.

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. Articulo publicado: *Problemas de rutas abiertas de vehículos con restricciones de capacidad heterogéneas*, Revista I+D (Innovación mas 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] Stefan Voß. *Meta-heuristics: The State of the Art*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[2] C. Martínez. *Metaheurísticas híbridas aplicadas al problema de ruteo de arcos capacitados*. PhD thesis, Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires, 2011.

[3] Mostepha R Khouadjia, Briseida Sarasola, Enrique Alba, Laetitia Jourdan, and El-Ghazali Talbi. A comparative study between dynamic adapted pso and vns for the vehicle routing problem with dynamic requests. *Applied Soft Computing*, 12(4):1426–1439, 2012.

[4] Yi Mei, Ke Tang, and Xin Yao. Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, 15(2):151–165, 2011.

[5] Bruce L Golden and Richard T Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

[6] Sanne Wøhlk. A decade of capacitated arc routing. In *The vehicle routing problem: latest advances and new challenges*, pages 29–48. Springer, 2008.

[7] Sue Conger, MaryAnne Winniford, and Lisa Erickson-Harris. Service management in operations. 2008.

[8] Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[9] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[10] M Kiuchi, Y Shinano, R Hirabayashi, and Y Saruwatari. An exact algorithm for the capacitated arc routing problem using parallel branch and bound method. In *Spring national conference of the operational research society of Japan*, pages 28–29, 1995.

[11] Enrique Benavent, Vicente Campos, Angel Corberán, and Enrique Mota. The capacitated arc routing problem: lower bounds. *Networks*, 22(7):669–690, 1992.

[12] Horst Sachs, Michael Stiebitz, and Robin J Wilson. An historical note: Euler's königsberg letters. *Journal of Graph Theory*, 12(1):133–139, 1988.

[13] Mei-Ko Kwan. Graphic programming using odd or even points. *Chinese Math*, 1(273-277):110, 1962.

[14] CS Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974.

[15] Jack Edmonds and Ellis L Johnson. Matching, euler tours and the chinese postman. *Mathematical programming*, 5(1):88–124, 1973.

[16] Vittorio Maniezzo. Algorithms for large directed carp instances: urban solid waste collection operational support. *UBLCS Techni-cal Report Series, Bolonha, Italy: University of Bolonha*, page 27, 2004.

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

[18] Ramón Gallego, Antonio Escobar, and Eliana Toro. Técnicas metaheurísticas de optimización. *Pereira: Textos Universitarios Universidad Tecnológica de Pereira*, 2008.

[19] Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763, 2010.

[20] Ryuichi Hirabayashi, Y Saruwatari, and N Nishida. Tour construction algorithm for the capacitated arc routing-problems. *Asia-Pacific Journal of Operational Research*, 9(2):155–175, 1992.

[21] José M Belenguer and Enrique Benavent. A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30(5):705–728, 2003.

[22] Bruce L Golden, James S DeArmon, and Edward K Baker. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, 10(1):47–59, 1983.

[23] Gündüz Ulusoy. The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22(3):329–337, 1985.

[24] Alain Hertz, Gilbert Laporte, and Michel Mittaz. A tabu search heuristic for the capacitated arc routing problem. *Operations research*, 48(1):129–135, 2000.

[25] Luc Chapleau, Jacques A Ferland, Guy Lapalme, and Jean-Marc Rousseau. A parallel insert method for the capacitated arc routing problem. *Operations Research Letters*, 3(2):95–99, 1984.

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

[27] Fred Glover. Tabu searchpart i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[28] Alain Hertz, Gilbert Laporte, and Michel Mittaz. A tabu search heuristic for the capacitated arc routing problem. *Operations research*, 48(1):129–135, 2000.

[29] Alain Hertz and Michel Mittaz. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation science*, 35(4):425–434, 2001.

[30] Christos Voudouris. *Guided local search for combinatorial optimisation problems.* PhD thesis, University of Essex, 1997.

[31] Philippe Lacomme, Christian Prins, and Alain Tanguy. First competitive ant colony scheme for the carp. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 426–427. Springer, 2004.

[32] Xin Deng, Zhengyu Zhu, Yong Yang, Xiaohua Li, Yunyan Tian, and Mengshuang Xia. A genetic algorithm for the capacitated arc routing problem. In *Automation and Logistics, 2007 IEEE International Conference on*, pages 1551–1556. IEEE, 2007.

[33] Pablo Moscato and Carlos Cotta. Una introducción a los algoritmos memeticos. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 7(19):131–148, 2003.

[34] Thomas A Feo and Mauricio GC Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71, 1989.

[35] José Fernando Gonçalves and Mauricio GC Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[36] Cristian Martinez, Irene Loiseau, Mauricio GC Resende, and S Rodriguez. Brkga algorithm for the capacitated arc routing problem. *Electronic Notes in Theoretical Computer Science*, 281:69–83, 2011.

[37] Patrick Beullens, Luc Muyldermans, Dirk Cattrysse, and Dirk Van Oudheusden. A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147(3):629–643, 2003.

[38] Karl F Doerner, Richard F Hartl, Vittorio Maniezzo, and Marc Reimann. Applying ant colony optimization to the capacitated arc routing problem. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 420–421. Springer, 2004.

[39] Mohamed Reghioui, Christian Prins, and Nacima Labadi. Grasp with path relinking for the capacitated arc routing problem with time windows. In *Workshops on Applications of Evolutionary Computation*, pages 722–731. Springer, 2007.

[40] Nacima Labadi, Christian Prins, and Mohamed Reghioui. Grasp with path relinking for the capacitated arc routing problem with time windows. In *Advances in computational intelligence in transport, logistics, and supply chain management*, pages 111–135. Springer, 2008.

[41] José Brandão and Richard Eglese. A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 35(4):1112–1126, 2008.

[42] Joaquín Bautista, Elena Fernández, and Jordi Pereira. Solving an urban waste collection problem using ants heuristics. *Computers & Operations Research*, 35(9):3020–3033, 2008.

[43] Ke Tang, Yi Mei, and Xin Yao. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 13(5):1151–1166, 2009.

[44] Yi Mei, Ke Tang, and Xin Yao. Improved memetic algorithm for capacitated arc routing problem. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 1699–1706. IEEE, 2009.

[45] Matthew JW Morgan and Christine L Mumford. A weight-coded genetic algorithm for the capacitated arc routing problem. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 325–332. ACM, 2009.

[46] Liang Feng, Yew-Soon Ong, Quang Huy Nguyen, and Ah-Hwee Tan. Towards probabilistic memetic algorithm: An initial study on capacitated arc routing problem. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–7. IEEE, 2010.

[47] Haobo Fu, Yi Mei, Ke Tang, and Yanbo Zhu. Memetic algorithm with heuristic candidate list strategy for capacitated arc routing problem. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[48] Luís Santos, João Coutinho-Rodrigues, and John R Current. An improved ant colony optimization based algorithm for the capacitated arc routing problem. *Transportation Research Part B: Methodological*, 44(2):246–266, 2010.

[49] Xiaomeng Chen. Maens+: A divide-and-conquer based memetic algorithm for capacitated arc routing problem. In *Computational Intelligence and Design (ISCID), 2011 Fourth International Symposium on*, volume 1, pages 83–88. IEEE, 2011.

[50] Tiantang Liu, Zhibin Jiang, and Na Geng. A memetic algorithm with iterated local search for the capacitated arc routing problem. *International Journal of Production Research*, 51(10):3075–3084, 2013.

[51] Fábio Luiz Usberti, Paulo Morelato França, and André Luiz Morelato França. Grasp with evolutionary path-relinking for the capacitated arc routing problem. *Computers & Operations Research*, 40(12):3206–3217, 2013.

[52] Min Liu, Hemant Kumar Singh, and Tapabrata Ray. Application specific instance generator and a memetic algorithm for capacitated arc routing problems. *Transportation Research Part C: Emerging Technologies*, 43:249–266, 2014.

[53] Zhurong Wang, Haiyan Jin, and Manman Tian. Rank-based memetic algorithm for capacitated arc routing problems. *Applied Soft Computing*, 37:572–584, 2015.

[54] Yuning Chen, Jin-Kao Hao, and Fred Glover. A hybrid metaheuristic approach for the capacitated arc routing problem. *European Journal of Operational Research*, 253(1):25–39, 2016.

[55] Gianpaolo Ghiani, Francesca Guerriero, Gennaro Improta, and Roberto Musmanno. Waste collection in southern italy: solution of a real-life arc routing problem. *International Transactions in Operational Research*, 12(2):135–144, 2005.

[56] Anita Amberg, Wolfgang Domschke, and Stefan Voß. Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *European Journal of Operational Research*, 124(2):360–376, 2000.

[57] Feng Chu, Nacima Labadi, and Christian Prins. A scatter search for the periodic capacitated arc routing problem. *European Journal of Operational Research*, 169(2):586–605, 2006.

[58] Fábio Luiz Usberti, Paulo Morelato França, and André Luiz Morelato França. The open capacitated arc routing problem. *Computers & Operations Research*, 38(11):1543–1555, 2011.

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

[60] Mauricio GC Resende and Celso C Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of metaheuristics*, pages 283–319. Springer, 2010.

[61] Julián Aráoz, Elena Fernández, and Carles Franquesa. Grasp and path relinking for the clustered prize-collecting arc routing problem. *Journal of Heuristics*, 19(2):343–371, 2013.

[62] Abdesslem Layeb, Meryem Ammi, and Salim Chikhi. A grasp algorithm based on new randomized heuristic for vehicle routing problem. *CIT. Journal of Computing and Information Technology*, 21(1):35–46, 2013.

[63] Yannis Marinakis. Multiple phase neighborhood search-grasp for the capacitated vehicle routing problem. *Expert Systems with Applications*, 39(8):6807–6815, 2012.

[64] Amir Salehipour, Kenneth Sörensen, Peter Goos, and Olli Bräysy. Efficient grasp+ vnd and grasp+ vns metaheuristics for the traveling repairman problem. *4OR: A Quarterly Journal of Operations Research*, 9(2):189–209, 2011.

[65] Joseph Gallart Suárez and Manuel Tupia Anticona. Solving the capacitated vehicle routing problem and the split delivery using grasp metaheuristic. In *IFIP International Conference on Artificial Intelligence in Theory and Practice*, pages 243–249. Springer, 2010.

[66] Christian Prins, Caroline Prodhon, and Roberto Wolfler Calvo. Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR: A Quarterly Journal of Operations Research*, 4(3):221–238, 2006.

[67] Yannis Marinakis, Athanasios Migdalas, and Panos M Pardalos. Expanding neighborhood grasp for the traveling salesman problem. *Computational Optimization and Applications*, 32(3):231–257, 2005.

[68] C Prins and R Wolfler Calvo. A fast grasp with path relinking for the capacitated arc routing problem. In *Proceeding of INOC*, pages 289–95, 2005.

[69] Marcelo Prais and Celso C Ribeiro. Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176, 2000.

[70] Gilbert Laporte, Michel Gendreau, J-Y Potvin, and Frédéric Semet. Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285–300, 2000.

[71] 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.

[72] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[73] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

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

[75] 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.

[76] 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.

[77] Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009.

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