MODELACIÓN Y DISEÑO DE UN SISTEMA DE REDES DE DISTRIBUCIÓN CON FLOTA HETEROGÉNEA EMPLEANDO LA METAHEURÍSTICA DE BÚSQUEDA TABÚ

DUVÁN EDUARDO PUENAYÁN T.

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INDUSTRIAL
INGENIERIA INDUSTRIAL
CALI
2013

MODELACIÓN Y DISEÑO DE UN SISTEMA DE REDES DE DISTRIBUCIÓN CON FLOTA HETEROGÉNEA EMPLEANDO LA METAHEURÍSTICA DE BÚSQUEDA TABÚ

DUVÁN EDUARDO PUENAYÁN T.

Trabajo de Grado para Optar por el Título de:
Ingeniero Industrial

DIRECTOR
JULIO CÉSAR LONDOÑO., MSc

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INDUSTRIAL
INGENIERIA INDUSTRIAL
CALI
2013

ESCUELA DE INGENIERÍA INDUSTRIAL
PROGRAMA DE INGENIERÍA INDUSTRIAL

AUTOR:

DUVÁN EDUARDO PUENAYÁN T.

TITULO:

MODELACIÓN Y DISEÑO DE UN SISTEMA DE REDES DE DISTRIBUCIÓN CON FLOTA HETEROGÉNEA EMPLEANDO LA METAHEURÍSTICA DE BÚSQUEDA TABÚ

DESCRIPTORES, MATERIAS O TEMAS QUE TRATA EL TRABAJO DE GRADO:

Logística, Transporte, Metaheurísticas, Problema del Ruteo de Vehículos con Flota Heterogénea (MFVRP), Búsqueda Tabú, Búsqueda Tabú Granular.

NOTA DE ACEPTACIÓN

_____

_____

Presidente del Jurado: _____

Jurado: _____

Jurado: _____

Santiago de Cali, 06 de Diciembre de 2013

AGRADECIMIENTOS

Primero que todo quiero agradecer a Dios por haberme permitido culminar esta etapa de mi vida. A mis padres que siempre estuvieron ahí para apoyarme, brindarme consejos y alientos cuando más lo necesitaba. A mis hermanas por haberme ayudado durante estos 5 años que estuve en la Universidad. A mi director de tesis Julio Cesar Londoño por el acompañamiento durante este proceso final de la carrera. Por último quisiera agradecer a mis compañeros de estudio, los cuales hicieron que esta etapa de mi vida fuera más divertida y memorable.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Debido a los cambios causados por la globalización, el acceso a nuevos mercados, culturas y tecnologías, las empresas se ven obligadas a tomar cada vez decisiones más complejas, siempre buscando lograr un alto grado de competitividad. Un factor esencial para la competitividad de las compañías, ha sido el manejo de la logística y su relación con el nivel de servicio al cliente. En este sentido la misión de la Logística es la de proveer los productos o servicios correctos, en el lugar correcto, en el momento adecuado, en la condición deseada, obteniendo la máxima contribución para la organización (Ballou, 1999). Sobre este tema, las empresas han dedicado gran parte de su esfuerzo en optimizar tales áreas, pues anteriormente dicho mejoramiento se hacía innecesario debido al pequeño número de empresas que conformaban la competencia.

Una de las decisiones de mayor transcendencia que la empresa debe afrontar, está relacionada con el diseño y gestión de la distribución física, ésta son todas aquellas actividades encaminadas a la planificación, implementación y control de un flujo creciente de materias primas, recursos de producción y productos finales desde el punto de origen al de consumo (Ballou, 1999). Entre estas actividades se encuentran el servicio al cliente, la distribución y el transporte, la previsión de la demanda, el control de inventarios, el manejo de mercancías, y el tratamiento de las mercancías devueltas entre otras.

La importancia de la eficacia y la eficiencia en la gestión de la distribución adquieren su verdadera magnitud cuando se consideran los costos, en este sentido los principales elementos de los costos de la distribución física son el transporte (37%), el control de existencias (22%), el almacenamiento (21%) y otros como la recepción de órdenes, el servicio al cliente, la distribución y la administración (20%) (Kotler, 1991).

Teniendo en cuenta que el porcentaje de los costos incurridos en los sistemas de transporte es el de mayor participación, es posible considerar que las decisiones en los sistemas de distribución son fundamentales y presentan una potencial ventaja competitiva que actualmente concentra grandes desafíos de gestión para las empresas de todos los sectores. Es ésta necesidad de transportar bienes la que lleva a las investigaciones a enfocarse en el diseño y organización de mejores sistemas de distribución, cuya finalidad será la de cumplir a cabalidad con los

requerimientos del cliente, pero siempre considerando de vital importancia la reducción de los costos y el mejoramiento de la programación de rutas.

En este contexto, la programación y diseño de rutas en los sistemas de transporte son fundamentales para su buen funcionamiento. Centrándose en la dificultad de la distribución física se puede establecer que el problema de distribuir productos desde ciertos depósitos a sus usuarios finales, juega un papel central en la gestión de algunos sistemas logísticos, y su adecuada planificación puede significar considerables ahorros (Toth & Vigo, 2000). Con base en la anterior afirmación la contrariedad al momento de transportar bienes se presenta cuando se debe determinar el tipo de recursos a utilizar, la cantidad y las rutas a seguir, lo que se denomina problema de ruteo, y es tratado en la literatura como el Problema de Ruteo de Vehículos (VRP[1]).

Sobre la base de los elementos anteriores, en este trabajo de grado se abordó el diseño y programación de rutas de vehículos para una flota de camiones con diferentes capacidades y costos, esta variante del VRP se conoce como MFVRP o problema de ruteo de vehículos con flota heterogénea. La metaheurística empleada para solucionar el problema anterior fue la muy reconocida Búsqueda Tabú a la cual se le desarrolló una técnica efectiva de estrategias de listas de candidatos, el enfoque propuesto llamado Búsqueda Tabú Granular (GTS) (Toth & Vigo, 2003), se basa en el uso de vecindarios granulares, los cuales incluyen un pequeño número de movimientos "prometedores", que se evalúan en menos tiempo que los completos, permitiendo así encontrar soluciones en menos tiempo que el algoritmo de Búsqueda Tabú original.

Una vez implementado el algoritmo GTS se aplicó a un caso estudio el cual contaba con 79 clientes dispersos por toda la ciudad de Cali, una flota de camiones de distintas capacidades y unas demandas por atender. Los resultados obtenidos al aplicar el algoritmo de Búsqueda Tabú Granular cumplieron con las expectativas ya que se logró disminuir los costos de transporte representados en menores distancias recorridas en cada ruta, una mejor utilización de la capacidad de los vehículos y una reducción en los camiones empleados para atender a los clientes.

[1] Vehicle Routing Problem.

PLANTEAMIENTO DEL PROBLEMA

En la actualidad, la dificultad de distribuir productos a partir de un depósito o punto de origen a una cantidad de clientes con una demanda por atender, juega un papel importante en las empresas ya que programar adecuadamente estos envíos puede significar considerables ahorros logísticos y sobre todo en costos como el consumo de combustible, horas hombre, entre otros; lo que ayudaría a generar una mejor rentabilidad para las empresas. De hecho, el proceso de transporte involucra todas las etapas de la producción y los sistemas de distribución y representa un componente relevante (por lo general de 10% al 20%) del costo final de las mercancías (Toth & Vigo, 2000).

Todas las empresas que tienen la necesidad de transportar sus bienes desde el punto de origen hasta los clientes finales, se ven afectadas por la gran gama de factores a considerar, cuando se busca la mejor manera de llevar productos hacia los clientes empleando vehículos de carga, lo cual genera una contrariedad en cuanto al empleo de recursos y como consecuencia lógica en los costos que se incurren al momento de ejecutar las decisiones tomadas. Por lo anterior surge el problema de ruteo de vehículos, el cual puede describirse como sigue: “Dada una flota de vehículos con capacidad uniforme, un depósito común y la demanda de varios clientes, se debe encontrar el conjunto de rutas con un costo total mínimo para dar servicio a todas las demandas” (Cordeau, Laporte, Savelsbergh, & Vigo, 2007).

Aun sin considerar el grado de dificultad presente en el ruteo de vehículos, muchas empresas sostienen que el objetivo último de la distribución física es obtener las mercancías necesarias, llevarlas a su destino en el tiempo oportuno y al costo más bajo posible. Sin embargo, tal y como afirma Kotler (1991), no existe ningún sistema de distribución que pueda, simultáneamente, maximizar el servicio al cliente y minimizar los costos de distribución, puesto que lo primero supone un elevado costo de existencias, un transporte rápido y múltiples almacenes, factores que incrementan los costos. Es por ello que los inconvenientes relacionados con el diseño de rutas de vehículos sigue siendo un problema vigente que muchos autores continúan abordando con el fin de aportar mejores soluciones cada día.

Los inconvenientes en la distribución física de mercancías constituyen un conjunto variado y complejo de casos que algunos autores han intentado agrupar atendiendo a sus características más relevantes. Esta simplificación de la realidad

permite la adopción de modelos matemáticos los cuales facilitan los procesos de decisión que atañen a las empresas cuya necesidad es la de transportar sus productos hacia los clientes, por lo cual existen múltiples variantes del VRP, por ejemplo en los problemas del tipo Pickup and Delivery[2] se deben recoger bienes de un determinado cliente para ser llevada a otro distinto en la misma ruta. En los llamados VRP with Backhauls[3] debido a restricciones de carga y descarga y de la forma que se acomodan los bienes en los vehículos, se deben realizar todas las entregas antes de continuar con las recolecciones (Gendreau, Hertz, & Laporte, 1994).

En el presente proyecto se trabajará exclusivamente con el Problema de Ruteo de Vehículos con Flota Heterogénea (MFVRP)(Shen & Liu, 1999), este es un VRP en el que se suponen vehículos con distintas capacidades, para lo cual hay que determinar el tipo de vehículo y la ruta a seguir con el fin de atender a un conjunto de clientes, por lo que es necesario considerar las capacidades en la ruta que seguirá cada recurso, ya que un camión más grande podrá realizar una ruta más larga o que tenga mayor concentración de demanda. De manera general los objetivos considerados para el problema de ruteo de vehículos son; Minimización de los costos de transporte totales, dependiendo de la distancia total recorrida (o en el tiempo de viaje total); Disminución de los costos fijos asociados con los vehículos utilizados (y los operadores correspondientes); Minimización de la cantidad de vehículos (u operarios) necesarios para atender a todos los clientes y finalmente buscar un equilibrio de las rutas, el tiempo de viaje y la carga del vehículo (Toth & Vigo, 2000).

Para la solución de este problema de optimización combinatoria existen algoritmos de solución exactos y algoritmos de solución aproximados. Los primeros tratan de asegurar la obtención de la solución óptima del problema, a riesgo de emplear un tiempo de computación excesivo, a veces no disponible, por lo cual no los hace viable desde el punto de vista práctico (Daza, Montoya, & Narducci, 2009). Los algoritmos de solución aproximados son métodos que aportan de forma aproximada soluciones satisfactorias, entre ellos se encuentran las heurísticas y metaheurísticas, las cuales dan soluciones con valores cercanos a un óptimo en tiempos computacionales razonables (Vélez & Montoya, 2007). De manera general las metaheurísticas aportan soluciones a problemas difíciles de optimización combinatoria, en donde las heurísticas no son eficientes, es por ello que son la mejor opción para abordar el problema de ruteo de vehículos (Olivera,

[2] Recogida y Entrega.
[3] Con viajes de regreso.

2004). Para tratar el MFVRP propuesto en este trabajo se empleará la metaheurística *Tabú Search* cuya diferencia principal de otras metaheurísticas radica en que posee memoria, característica que la hace interesante para un trabajo de investigación.

A pesar de las simplificaciones propuestas por muchos autores en cuanto al diseño y programación de rutas, las situaciones reales se escapan a una clasificación tan simple, por ejemplo la legislación sectorial, las condiciones atmosféricas, etc., suman restricciones que alejan los esquemas teóricos estudiados por los investigadores de la realidad cotidiana. Lo anterior aumenta la complejidad en el diseño y programación de rutas, donde los operadores de transporte deben tomar decisiones continuamente, tanto en el día a día como a mediano y largo plazo. En este sentido, no es de extrañar que los responsables de estos asuntos simplifiquen al máximo las contrariedades y utilicen procedimientos particulares para despachar sus vehículos basándose, en multitud de ocasiones en la experiencia de errores anteriores. Debido a estas razones, las consecuencias del problema de ruteo en las empresas se siguen presentando, generando altos costos, ventas perdidas y pérdidas de clientes. Según Gómez (2006), las ventas perdidas por no cumplir pedidos perfectos o porque los productos están agotados en los puntos de venta pueden llegar a $1 billón de pesos a pesos del año 2006, para el caso de Colombia.

Finalmente considerando la manera como en la actualidad se realizan estas operaciones de carácter trascendental, existe una ventana de oportunidades, encaminada al desarrollo de metodologías sencillas, buscando la rentabilidad de las empresas. En este sentido la investigación pretende responder la pregunta ¿Qué resultados se pueden obtener abordando la solución, evaluación y análisis del problema de ruteo de vehículos desde la perspectiva de los costos de transporte y los beneficios para la compañía?

OBJETIVOS

Objetivo General

Construir un modelo que permita hallar una solución al problema de ruteo de vehículos con flota heterogénea basado en la técnica *Tabú Search,* con el fin de minimizar los costos asociados al transporte.

Objetivo Específicos

- ✓ Estudiar la aplicabilidad, ventajas y desventajas de las herramientas existentes para la solución del problema de ruteo de vehículos con flota heterogénea, mediante la revisión bibliográfica pertinente.

- ✓ Documentar el caso de estudio por medio de la recolección de información relacionada con la ubicación y demanda de los clientes, además de establecer las restricciones que requiera el diseño del sistema de redes.

- ✓ Aplicar el modelo desarrollado para demostrar los beneficios que conlleva la herramienta en las situaciones reales, mediante el empleo de datos proporcionados por empresas del sector.

- ✓ Analizar la aplicabilidad del modelo para el ruteo de vehículos a través de indicadores de medición que permitan determinar el impacto de la solución en relación a los costos de transporte.

JUSTIFICACIÓN

Hoy en día la programación y diseños de rutas para la distribución física de productos se sigue constituyendo como un serio problema, que en la mayoría de los casos no es resuelto de manera eficaz, afectando las ganancias de las compañías y obligando a las mismas a enfrentar retos dinámicos sin las suficientes herramientas. Las metaheurísticas ayudan a simular estrategias eficientes que conllevan a encontrar y aplicar modelos operativos que aportan proactivamente en el sistema donde interactúan los elementos compañía - cliente.

Desde el punto de vista de las empresas, una deficiente programación y diseño de rutas genera un alto costo logístico, en cuanto al empleo de sus recursos ya que según Ballou (1999), estos costos oscilan entre menos del 4% sobre las ventas en aquellas empresas que producen y distribuyen mercancías de alto valor, hasta más de un 32% en aquellas otras que lo hacen en las de bajo valor, además los costos de transporte representan entre una tercera y dos terceras partes del total de costos logísticos. Una deficiente programación de rutas conlleva a pérdidas en cuanto a sus ventas, debido a que los clientes no son atendidos en el momento oportuno. En Colombia, las ventas perdidas por no cumplir pedidos perfectos o porque los productos están agotados en los puntos de venta pueden llegar a $1 billón de pesos a pesos del año 2006, las dos variables (Pedidos perfectos y agotados) son prioritarias al medir la eficiencia en la operación logística (Gómez, 2006).

Por otro lado los sistemas de transporte terrestre son un factor clave para el crecimiento económico, dichos sistemas representan un gran porcentaje al PIB del país, lo cual se ve reflejado en los niveles de crecimiento y de aceptación en los mercados internacionales; en el caso colombiano, la carga que se transporta por vía terrestre es aproximadamente el 80% del total, a través de una red insuficiente y con limitaciones (Mintransporte, 2011). La contribución de los servicios de transporte al PIB, para el período 2010, fue de 18.089 miles de millones de pesos, lo que representa un 4.26 % del PIB nacional, del cual el 76%, es decir, 13.577 miles de millones de pesos representa el porcentaje de participación de los servicios de transporte terrestre (Mintransporte, 2012). Es aquí donde se ve reflejada la importancia de tratar los problemas de ruteo de vehículos, con el fin de buscar la eficiencia en función de los costos, considerando como beneficiario de manera indirecta la economía del país.

Finalmente el diseño y gestión de la distribución física conlleva una multitud de decisiones que deben abordarse con métodos que permitan al gestor identificar sus oportunidades de mejora, reduciendo costos, disminuyendo el nivel de inversión necesario y mejorando el servicio hacia los clientes. Incluso el recorte de una pequeña fracción de los costos puede aflorar enormes ahorros económicos y una reducción de los impactos medioambientales ocasionados por la polución y el ruido, además de incrementar significativamente la satisfacción de los requerimientos de los clientes. Esos potenciales ahorros justifican en gran medida la utilización de técnicas de investigación operativa como facilitadoras de la planificación.

1. MODELOS DE DISTRIBUCIÓN FÍSICA Y TRANSPORTE.

El Problema de Ruteo de Vehículos se encuentra en el núcleo de la gestión de la distribución. A él se enfrentan cada día miles de empresas y organizaciones dedicadas a la entrega y recolección de mercancías o personas. En términos simples, el objetivo es determinar un conjunto de rutas con el mínimo costo total que pueden satisfacer varias demandas dispersas geográficamente, una flota de vehículos situados en uno o más depósitos están disponibles para satisfacer las demandas.

La programación y diseño de rutas de vehículos consta de tres elementos fundamentales (Toth & Vigo, 2000), el primero son los clientes, cada cliente tiene cierta demanda que deberá ser satisfecha por algún vehículo, en la vida real existen características típicas que diferencian a un grupo de clientes, como por ejemplo, el tipo de demanda, periodos de tiempo durante el cual el cliente puede ser servido, el nivel de prioridad, etc. El segundo elemento son los vehículos, en los cuales se realiza el transporte de mercancías, algunas de sus características se refieren a la capacidad del vehículo, al costo asociado con la utilización del mismo, o a su centro de distribución (CD), ya que su punto de partida y llegada podría ser o no ser el mismo. Finalmente, como tercer elemento se encuentran los centros de distribución o depósitos, los cuales pueden albergar tanto a los vehículos como los bienes a distribuir. Teniendo en cuenta lo anterior es posible profundizar más en algunos de los modelos de distribución física más conocidos.

Según Vidal (2010), los problemas de ruteo se pueden clasificar en dos grandes grupos, los problemas con origen y destino diferentes y los problemas con el mismo origen y destino, los últimos presentan mayor interés ya que en torno a este grupo se desarrolla el actual trabajo. Dentro de esta categoría el problema más básico de todos es el problema del viajero de negocios, conocido como TSP[4].

1.1 El Problema Del Viajero De Negocios (TSP)

El problema del viajero de negocios es muy sencillo de plantear. Un agente viajero parte desde una ciudad de origen para visitar exactamente un conjunto de n ciudades, una sola vez cada una y regresar a la ciudad origen. A este recorrido se le llama un tour. El objetivo es minimizar la distancia total recorrida en el tour.

[4] Traveling Salesman Problem.

De manera general, este problema se caracteriza porque no suele haber un depósito, no hay demanda asociada a los clientes y tampoco hay restricciones temporales. Según (Dantzig, Fulkerson, & Johnson, 1954) El problema puede formularse como sigue:

Algoritmo 1.1

$x_{ij} = 1, si\ el\ arco\ (i,j)\ pertenece\ a\ la\ ruta; 0\ de\ lo\ contrario.$

$c_{ij} = costo\ del\ nodo\ i\ al\ nodo\ j.$

$V = Conjunto\ de\ vertices.$

$S = Subconjunto\ de\ V.$

$E = Conjunto\ de\ arcos\ no\ direccionados.$

$$\min \sum_{(i,j)\in E} c_{ij}\, x_{ij} \qquad (1.1)$$

$$\text{s.a} \sum_{j\in\Delta+(i)} x_{ij} = 1 \qquad \forall i \in V \qquad (1.2)$$

$$\sum_{i\in\Delta-(j)} x_{ij} = 1 \qquad \forall j \in V \qquad (1.3)$$

$$\sum_{i\in S, j\in\Delta+(i)/S} x_{ij} \geq 1 \qquad \forall S \subset V \qquad (1.4)$$

$$x_{ij} \in \{0,1\} \qquad \forall\ (i,j)\ \in E$$

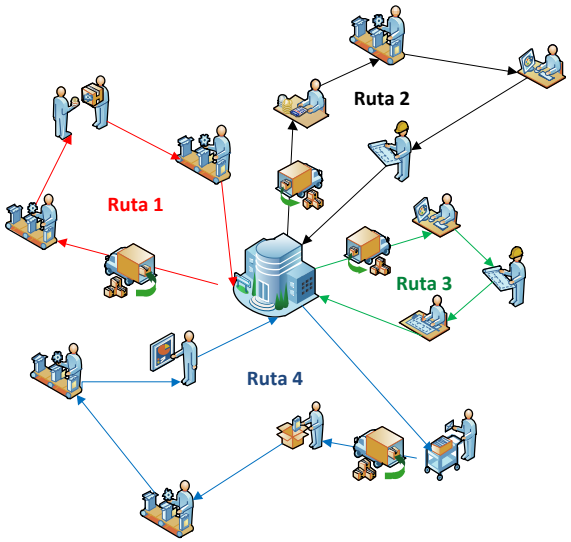
Las variables binarias x_{ij} indican si el arco (i,j) es utilizado en la solución. La función objetivo (1.1) establece que el costo total de la solución es la suma de los costos de los arcos utilizados. Las restricciones 1.2 y 1.3 indican que la ruta debe llegar y abandonar cada nodo exactamente una vez. Finalmente, las restricciones 1.4 son llamadas restricciones de eliminación de sub-tours e indican que todo subconjunto de nodos S debe ser abandonado al menos una vez (Olivera, 2004).

1.2 El Problema De Ruteo De Vehículos (VRP)

El problema del ruteo de vehículos es mucho más complejo que el problema del TSP. En realidad un VRP consta de múltiples problemas del TSP, pero con restricciones y características adicionales que pueden volver extremadamente complejo el problema. En este sentido, el VRP hace necesario seleccionar el conjunto de clientes para que sean atendidos separadamente por cada vehículo y después determinar el orden de servicio de los mismos, teniendo en cuenta que la suma de las demandas de cada cliente, debe ser menor o igual a la capacidad de

carga de cada vehículo. En la figura 1.1 se presenta una descripción gráfica para un VRP.

Figura 1.1 El problema de Ruteo de Vehículos



Fuente: Adoptado de Vidal (2010), pág. 422.

El problema de ruteo, en su forma más simple, se puede expresar de la siguiente forma (Vidal, 2010):

$$\boldsymbol{Minimizar}: \ Número\ de\ rutas, Vehículos, Distancia\ viajada, Costo\ total\ de\ las\ rutas$$
$$\boldsymbol{Maximizar}: \ Utilización\ de\ los\ Vehículos$$
$$\boldsymbol{Sujeto\ a}: \ Demandas, Restricciones\ de\ longitud\ y\ velocidad, Balanceo\ de\ rutas, Distancia\ viajada, \ Restricciones\ de\ tiempo$$

La función objetivo de un problema de ruteo puede ser compleja. Por supuesto que el objetivo primordial es la minimización del costo total de todas las rutas, el cual puede estar implicado por la minimización de la distancia total y del número de rutas o vehículos utilizados (Vidal, 2010). Por otra parte el VRP debe cumplir varias restricciones operativas, que dependen de la naturaleza de las mercancías transportadas, de la calidad del nivel de servicio, de las características de los clientes y de los vehículos. Algunas restricciones operativas típicas son las siguientes: a lo largo de cada ruta, la carga actual del vehículo asociado no puede exceder su capacidad, los clientes servidos en una ruta pueden pedir sólo la entrega o la recogida de los bienes, o ambas posibilidades pueden existir, etc.

Toda la anterior diversidad y variabilidad de situaciones puede volver al VRP un problema altamente complejo, para el cual es imposible hallar soluciones óptimas (Yepes, 2002). A continuación se describirán brevemente algunas de las variantes básicas del VRP (Toth & Vigo, 2000) (Zanjirani, Rezapour, & Kardar, 2011):

1.2.1 El VRP con Restricciones de Capacidad (CVRP)

En su forma clásica el CVRP[5] se define como el diseño de una serie de rutas de mínimo costo; para atender un conjunto de clientes geográficamente dispersos, cada uno con demanda y localización conocidas, utilizando una flota de vehículos iguales (homogénea), visitando cada cliente una sola vez y cumpliendo con las restricciones de capacidad de los vehículos, además, cada ruta inicia y termina en un mismo lugar (Villegas, Lopera, & Sanchez, 2007).

1.2.2 El VRP con Restricciones de Capacidad y Distancia (DCVRP)

El DCVRP[6] es una variante de CVRP en la que se imponen tanto las restricciones de capacidad del vehículo como las restricciones de máxima distancia a recorrer. En este tipo de problemas, cada longitud de recorrido no debe exceder la distancia conocida con anterioridad. El objetivo es minimizar el número de vehículos usados, además de minimizar el costo de viaje total (Toth, 2010).

1.2.3 El VRP con Ventanas de Tiempo (VRPTW)

El VRPTW[7] es la extensión del CVRP en el cual restricciones de capacidad se imponen y cada cliente se asocia con un intervalo de tiempo en el cual debe ser servido, este intervalo se llama ventana de tiempo. El VRPTW consiste en encontrar una colección exacta de circuitos simples con un costo mínimo, y tal que cumpla con los supuestos para el CVRP además de que para cada cliente, el servicio se inicia dentro de la ventana de tiempo (Zanjirani et al., 2011).

[5] Capacitated VRP
[6] Distance-Constrained and Capacitated VRP
[7] VRP with Time Windows

1.2.4 El VRP con Viajes de Regreso (VRPB)

El VRP con viajes de regreso es la extensión del CVRP, donde los clientes pueden requerir o devolver algunos productos. En esta variante el cliente se divide en dos subgrupos. El primer subgrupo, contiene a los clientes que requieren una determinada cantidad de producto a entregar. El segundo subgrupo, contiene los clientes donde una cantidad determinada de producto debe ser recogida. En el VRPB[8], adiciona una restricción de precedencia entre los clientes a los cuales se les debe entregar productos y a los que se les debe recoger (Toth & Vigo, 2000).

Finalmente, como se puede apreciar, existe una gran cantidad de problemas derivados del VRP, las descritas anteriormente se conocen como las variantes básicas. Yepes (2002), elaboró una tabla en la cual se recogen algunos de los problemas de la asignación y programación de rutas hasta el momento. Según Zanjirani *et al.* (2011), existen otras familias del VRP que son de particular importancia para la investigación de operaciones y especialmente para las actividades de distribución física, algunas de ellas son; el VRP abierto (Open VRP); el VRP con Múltiples Depósitos (Multidepot VRP); el VRP con Flota Heterogénea (HVRP); entre otros.

El siguiente trabajo se centra en la familia de VRP's denominada Mixed Fleet VRP (MFVRP) o Heterogeneous Fleet VRP (HVRP) (Baldacci, Battarra, & Vigo, 2007), esta variante se caracteriza porque las demandas de los clientes son satisfechas con vehículos de diferentes capacidades, además de poseer costos fijos y costos variables por utilización.

1.3. El VRP Con Flota Heterogénea (MFVRP).

Aunque en la teoría a menudo se supone, la flota de vehículos de una compañía de camiones raramente es homogénea. Los vehículos difieren en sus equipos, capacidad de carga, edad y estructura de costos. (Dullaert et al., 2002). El MFVRP es un tipo diferente de VRP que se diferencia del clásico en el cual se trata con una flota heterogénea de vehículos con diferentes capacidades y costos, los costos pueden ser fijos por compra o renta del transporte y los costos variables se asignan por unidad de distancia recorrida en el trayecto. El costo de enrutamiento es la suma de los costos fijos y variables. El objetivo del MFVRP es determinar la

[8] VRP with Backhauls

combinación óptima de vehículos heterogéneos y su conjunto asociado de rutas posibles que minimizan el costo de enrutamiento, sujeto a las restricciones de los vehículos y los clientes.

Hay tres tipos de MFVRP en la literatura referenciados con distintos nombres, según la clasificación de Zanjirani *et al* se encuentra: El primer tipo el cual fue introducido por (Golden et al., 1984), este utiliza el mismo valor para los costos variables, independientemente del tipo de vehículo y tiene un número ilimitado de vehículos de cada tipo. A esta variante del MFVRP se le conoce como VRP con Tamaño de Flota Combinada (**FSMVRP**[9]) o VRP con Tamaño de Flota Compuesta.

El segundo tipo, propuesto por (Salhi et al., 1992), considera diferentes costos variables que dependen del tipo de vehículo y también tiene un número ilimitado de vehículos de cada tipo. Se le conoce como **VFM**[10] con Costo Variable por Unidad de Desplazamiento, sí se desea observar una formulación de este modelo se debe referenciar a (Osman & Salhi, 1996).

(Taillard, 1999), introduce el último tipo el cual se diferencia del segundo en que hay restricciones en el número de los diferentes vehículos disponibles de cada tipo, en la literatura se le llama VRP Heterogéneo (**HVRP**[11]), sí se desea observar una formulación de este modelo se debe referenciar a (Baldacci et al., 2007).

Debido a la complejidad de la MFVRP, algunos han hecho intentos para formularlo usando programación lineal entera mixta, pero ningún algoritmo exacto para el MFVRP ha sido desarrollado (Zanjirani et al., 2011). En la **Tabla 1.1** se presenta una recopilación hecha por Zanjirani *et al.* (2011) de algunos artículos que trabajan las variantes del MFVRP y el algoritmo que se empleó para solucionar dicho problema de ruteo. En esta tabla además aparece una variante del FSMVRP, en cuyo problema aparece la restricción de ventanas de tiempo, conocida como FSMVRP con Ventanas de Tiempo (FSMVRPTW[12]).
Existe otra clasificación de las distintas variantes del MFVRP presentada por Baldacci *et al.* (2007), este documento ofrece una visión general de los enfoques de la literatura para resolver el VRP heterogéneo. Según los autores, las diferentes variantes del problema han sido referidas en la literatura utilizando

[9] Fleet Size and Mixed VRP
[10] Vehicle Fleet Mix
[11] Heterogeneous VRP
[12] FSMVRP with Time Windows

distintos nombres, aunque hay una cierta homogeneidad al llamar VRP heterogéneo (HVRP) a las variantes con un **número limitado de vehículos**, y el VRP con Tamaño de Flota Combinada (FSMVRP) a las variantes con un **número ilimitado de vehículos**.

En este trabajo se empleará la nomenclatura literaria hecha por Zanjirani *et al*. (2011), para describir los tres tipos de variantes del MFVRP.

Tabla 1.1 Literatura del MFVRP.

Autor (es)	Año	Tipo	Algoritmo
Golden et al.	1984	FSMVRP	Heurística de Ahorros basada en el Método de Clarke -Wright, Procedimiento de 2 pasos
Gheysens et al.	1984	FSMVRP	Acercamiento enfocado en Castigos
Gheysens et al.	1986	FSMVRP	Método de Dos Estados
Salhi et al.	1992	VFM	Perturbación de ruta (**RPERT**) Procedimiento para diferentes costos variables
Salhi and Rand	1993	FSMVRP	Extension del Procedimiento **RPERT**
Osman and Salhi	1996	VFM	Versión modificada de **RPERT**, llamada **MRPERT**, permitiendo un proceso de búsqueda para reiniciar varias veces y producir varias soluciones
Taillard	1999	VFM	Método heurístico de Generación de Columnas
Liu and Shen	1999	FSMVRPTW	Heurística de Ahorros basada en Inserción
Renaud and Boctor	2002	FSMVRP	Algoritmo Basado en Barrido
Wassan and Osman	2002	FSMVRPTW	Nuevas variantes de Búsqueda Tabú combinada con conceptos de Búsqueda Tabú Reactiva, variables de vecindario, estructuras especiales de memoria de datos y funciones hash
Dullaert et al.	2002	FSMVRPTW	Heurística de inserción secuencial
Dell'Amico et al.	2006	FSMVRPTW	Algoritmos Heurísticos de inserción constructiva y metaheurísticas
Belfiore and Favero	2007	FSMVRPTW	Acercamiento de Búsqueda Dispersa
Braysy et al.	2008	FSMVRPTW	Metaheurística de Recocido Determinístico con Multiples Inicios
Lee et al.	2008	VFM	BúsquedaTabú y serie particionado
Brandao	2009	VFM	Algoritmo de Búsqueda Tabú Determinístico
Braysy et al.	2009	FSMVRPTW	Metaheurística de 3 Fases
Liu et al.	2009	FSMVRP	Heuristica Basada en Algoritmos Genéticos
Repoussis and Tarantilis	2009	FSMVRP	Acercamiento de Solución con Programación de Memoria Adaptativa

Fuente: Adaptada de Zanjirani *et al*. (2011), Pág. 140.

2. TÉCNICAS DE SOLUCIÓN PARA EL VRP CON FLOTA HETEROGÉNEA.

Debido a la dificultad intrínseca del VRP con flota heterogénea, ningún algoritmo de resolución exacto ha sido desarrollado, en este sentido, todos los enfoques de solución presentados hasta ahora en la literatura son algoritmos heurísticos y metaheurísticos. Además, por lo general, son adaptaciones o extensiones de los métodos propuestos en las últimas décadas para las variantes básicas del VRP, como el VRP y VRP con ventanas de tiempo (Baldacci et al., 2007).

En este capítulo se describirán brevemente los principales algoritmos de resolución aproximados empleados para solucionar el MFVRP, sus ventajas y desventajas, cubriendo de esta manera el primer objetivo específico de este trabajo de grado.

2.1 Heurísticos Para El MFVRP

Según Olivera (2004), las heurísticas son procedimientos simples que realizan una exploración limitada del espacio de búsqueda y dan soluciones de calidad aceptable en tiempos de cálculo generalmente moderados. Un algoritmo es heurístico cuando la solución se determina por medio de pruebas y ensayos repetitivos. Esta técnica consiste en crear candidatos de posibles soluciones acorde a un patrón dado, siendo sometidos a pruebas de acuerdo a un criterio de la solución. Si un candidato no es aceptado, se genera nuevamente el proceso el cual se encarga de crear un nuevo candidato.

A continuación se presentan algunas heurísticas utilizadas para solucionar el MFVRP, en ciertos casos los autores complementaron dos o más algoritmos para solucionar alguna de las variantes del VRP con flota heterogénea.

2.1.1 El Método de los Ahorros

Uno de los algoritmos más difundidos para el VRP es el Algoritmo de Ahorros de (Clarke & Wright, 1964). Este algoritmo parte de una solución inicial en la cual cada cliente aparece separadamente en una ruta y se realizan las uniones de

rutas siempre y cuando sea factible, satisfaciendo las características del problema (Olivera, 2004).

El resultado del método puede indicar que el máximo ahorro sea negativo, esto indica que la combinación de las rutas aumentaría la distancia recorrida pero disminuirá la cantidad de vehículos a utilizar ya que el número de vehículos tiene relación directa con el número de rutas requeridas.

Uno de los más importantes aportes para tratar con el MFVRP fue propuesto por Golden *et al* (1984), el cual sugirió 5 adaptaciones para el algoritmo de los ahorros de Clarke y Wright (1964).
El primero, denominado *CW*[13], es una adaptación directa del algoritmo original para el MFVRP. El segundo, denominado *ahorros combinados*, o *CS*[14], extiende el concepto de ahorro para incluir los costos fijos del vehículo. Tres variantes del enfoque *CS* también se han desarrollado. En el algoritmo de *ahorro de oportunidad optimista*, *OOS*[15], los ahorros de oportunidad asociados con un vehículo adicional se define como el costo del vehículo más pequeño que puede dar servicio a la totalidad de la capacidad no utilizada del nuevo vehículo. El cuarto método presentado, llamado heurística de *ahorros de oportunidad realista*, *ROS*[16], sugiere que el ahorro de oportunidades fomenta el uso de vehículos más grandes cuando parece rentable hacerlo. Por lo tanto, la oportunidad de ahorro no debe ser incluida en la fórmula del ahorro a menos que la combinación de dos sub-tours requiera el uso de un vehículo más grande. Finalmente, el último algoritmo, llamado *ROS-*γ, es una variante del algoritmo *ROS* donde γ se utiliza como un parámetro de forma en el cálculo de ahorros (Renaud & Boctor, 2002).

2.1.1.1 Algoritmo de Ahorros basado en Matching

Cuando en el Algoritmo de Ahorros se decide unir dos rutas r_i y r_j, se está descartando otras uniones posibles en las que participan r_i o r_j (porque i y j dejan de ser extremos en la nueva ruta). La unión que da el máximo ahorro podría, en algunos casos, hacer que las uniones que permanecen factibles no sean buenas. Elegir siempre el máximo ahorro es una estrategia demasiado voraz. En el

[13] Clarke y Wright
[14] Combined Savings
[15] Optimistic Opportunity Savings
[16] Realistic Opportunity Savings

Algoritmo de Ahorros Basado en Matching se decide la unión a realizar considerando como afecta ésta a las posibles uniones en iteraciones siguientes.

(Desrochers & Verhoog, 1991), propusieron un algoritmo basado en el método de los ahorros para el MFVRP. Su *algoritmo de ahorros basado en Matching* (MBSA[17]) se basa en la fusión de rutas sucesivas donde la mejor fusión se selecciona mediante la resolución de un problema Matching de peso[18]. Un número de variantes de este algoritmo se proponen en la que cada variante utiliza una fórmula de ahorro diferente.

Como se expuso anteriormente el algoritmo de los ahorros en su versión más general tiene la ventaja de ser sencillo al momento de aplicarlo, pero su desventaja se presenta en su propio funcionamiento, ya que su proceso es demasiado voraz y puede generar desventajas al momento de aplicarlo a instancias grandes, en donde sus respuestas quedan cortas frente a las soluciones obtenidas con metaheurísticas más sofisticadas (Marti, 2004). En este trabajo se empleará el método de ahorros en su versión más básica como solución inicial de la metaheurística empleada.

2.1.2 Métodos Asignar Primero - Rutear Después

Los métodos asignar primero y rutear después (*cluster first - route second*) proceden en dos fases. Primero se busca generar grupos de clientes, también llamados clúster, que estarán en una misma ruta en la solución final. Luego, para cada clúster se crea una ruta que visite a todos sus clientes. Las restricciones de capacidad son consideradas en la primera etapa, asegurando que la demanda total de cada clúster no supere la capacidad del vehículo. Por lo tanto, construir las rutas para cada clúster es un TSP que, dependiendo de la cantidad de clientes en él, se puede resolver en forma exacta o aproximada.

2.1.2.1 Heurística de Barrido o Sweep

En este método se traza una línea imaginaria desde el CD en cualquier dirección y se pone a rotar, por ejemplo, en el sentido de las manecillas del reloj. A medida

[17] Matching Based Savings Algorithm.
[18] Matching en un grafo es un conjunto de arcos que no tienen extremos en común. El Peso de un Matching es la suma de los pesos de sus arcos.

que la recta imaginaria, como un radar, va “barriendo” la zona de ruteo, va tocando paradas. Estas paradas se adicionan a la primera ruta y se sigue rotando la recta hasta que no se puedan adicionar más paradas al grupo por limitaciones de capacidad del vehículo. En este momento se cierra esta ruta, se continúa rotando la recta y se inicia el grupo para una segunda ruta. Así se sigue hasta que todas las paradas hayan sido barridas por la recta. Una vez los grupos estén constituidos, se procede a hacer su secuenciación mediante las técnicas conocidas del TSP.

Podríamos iniciar el barrido desde cualquiera de las n paradas o clientes y, por lo tanto, podríamos tener hasta n diferentes resultados con este método. Simplemente escogemos el conjunto de rutas que produzcan la mínima distancia total recorrida (Vidal, 2010).

Renaud y Boctor (2002), presentan una nueva heurística basada en el Barrido para el MFVRP. La heurística que proponen utiliza diferentes procedimientos para generar un gran conjunto de buenas rutas y luego se eligen aquellas que satisfacen las restricciones del problema con el menor costo, utilizando un algoritmo polinomial de conjuntos de partición. En concreto, la heurística propuesta utiliza cinco procedimientos subordinados llamados: Orden, 1-pétalo, 2 pétalos, Selección de Pétalos y Mejora.

2.1.2.2 Heurística de Asignación Generalizada de Fisher y Jaikumar

(Fisher & Jaikumar, 1981), proponen generar los clústers resolviendo un Problema de Asignación Generalizada (GAP) sobre los clientes. Primero se fijan K clientes semilla S_k con $k = 1,\ldots,K$ sobre la base de los cuales se construirán los clúster. En una segunda fase, se decide qué clientes asignar a cada uno de los clúster de modo de no violar la capacidad del vehículo resolviendo un GAP.

En (Gheysens, Golden, & Assad, 1984) y (Gheysens, Golden, & Assad, 1986), se presentaron dos heurísticas para el MFVRP, en una de ellas se empleó la heurística de Asignación Generalizada. En la primera heurística se incorpora las restricciones de capacidad del vehículo en la función objetivo, junto con los costos fijos de los vehículos y los costos variables de la ruta, mediante el uso de multiplicadores de penalización. El problema resultante se resuelve usando el

algoritmo (MGT[19] + Or-opt[20]) para diferentes valores de los multiplicadores de penalización.

La segunda heurística es un algoritmo de dos etapas. En la primera etapa, un procedimiento de límite inferior se utiliza para determinar la mezcla de vehículos, es decir, el número de vehículos de cada tipo que se utiliza. En la segunda etapa, el *Procedimiento de Asignación Generalizada* de Fisher y Jaikumar (1981) se utiliza para resolver el problema de ruteo restante para los vehículos obtenidos.

Al igual que el método de los ahorros, el método de barrido en su versión más general es un procedimiento sencillo de implementar que ofrece buenas respuestas, pero su desventaja principal también radica en su sencillez, puesto que la calidad de las soluciones no son las mejores, por tan motivo algunos autores buscan la manera de complementarlo con otras técnicas, agregando cierto grado de complejidad en su implementación. El algoritmo de barrido en su versión más general será implementado en este trabajo para que genere una solución inicial.

2.1.3 Procedimientos de Búsqueda Local

Una vez que se tiene una solución para el problema, se puede intentar mejorarla mediante algún procedimiento de búsqueda local. Para cada solución s se define un conjunto de soluciones vecinas $N(s)$. Un procedimiento de Búsqueda Local parte de una solución s, la reemplaza por una solución $s^* \in N(s)$ de menor costo y repite el procedimiento hasta que la solución no pueda ser mejorada. Al terminar, se obtiene una solución localmente óptima respecto a la definición de la vecindad (Olivera, 2004).

Dentro de los procedimientos de búsqueda local se han utilizado dos algoritmos para tratar con el MFVRP ellos son:

2.1.3.1 El operador Or-opt

Una versión reducida del algoritmo 3-opt es el algoritmo Or-opt (Or, 1976), que consiste en eliminar una secuencia de k clientes consecutivos de la ruta y

[19] Multiple Partition Giant Tour Algorithm
[20] Heurística Or-Opt de Or (1976).

colocarlos en otra posición de la ruta, de modo que permanezcan consecutivos y en el mismo orden. Primero se realizan las movidas con $k = 3$, luego con $k = 2$ y finalmente con $k = 1$. Si una ruta visita n clientes existen $O(n^2)$ de estas movidas (Olivera, 2004).

Como se mencionó anteriormente Gheysens *et al.* (1984, 1986), utilizaron el operador Or - Opt junto con el algoritmo MGT para solucionar el MFVRP.

2.1.3.2 GENI y GENIUS

Las inserciones generalizadas GENI[21] surgen dentro de un método de solución del TSP y tienen como principal característica que la inserción de un cliente en una ruta no necesariamente ocurre entre dos nodos adyacentes (Gendreau, Hertz, & Laporte, 1992).

GENIUS se obtiene mediante la realización de GENI después de una fase llamada post-optimización US[22] (Gendreau et al., 1999). El algoritmo de post-optimización GENIUS comienza considerando el primer cliente de la ruta: se lo elimina mediante *Unstringing* y se lo vuelve a insertar utilizando *Stringing*. Esto podría incrementar el costo de la solución. Si la solución es mejorada, se vuelve a comenzar con el primer cliente de la nueva ruta. Si no es mejorada, se repite el proceso con el segundo cliente de la nueva ruta. El proceso termina luego de eliminar e insertar el último cliente de la ruta.

Gendreau et al. (1999) propusieron una heurística basado en Tabú, usando la heurística llamada GENIUS desarrollada para resolver el problema del viajero de negocios propuesto por Gendreau et al. (1992) con un procedimiento de memoria adaptativa. En este trabajo la función de GENIUS consiste en eliminar arcos y crear otros nuevos, invirtiendo además el sentido de los caminos, para después evaluar estos movimientos y si se obtienen mejoras se construyen las nuevas rutas.

De manera general los procedimientos de búsqueda local son mejores o presentan una mayor calidad respecto a los algoritmos de la subsección 2.1.1 y 2.1.2 (Martí, 2003), lo cual se convierte en una ventaja, desafortunadamente los

[21] Generalized Insertions
[22] Unstringing y Stringing

métodos de búsqueda local presentan una miopía en su desempeño, puesto que una vez alcanzado un óptimo local, el algoritmo no podrá mejorar este valor mediante otro movimiento definido, acortando su búsqueda del óptimo global.

2.2 Metaheurísticos Para El MFVRP

Debido a que la mayoría de los problemas de optimización combinatoria se clasifican como difíciles (Wassan & Osman, 2002), la investigación se ha concentrado en desarrollar algoritmos de aproximación. Dentro de esta área, el término metaheurístico lo introdujo Glover (1986), al definir una clase de algoritmos de aproximación que combinan heurísticos tradicionales con estrategias eficientes de exploración del espacio de búsqueda (Vélez & Montoya, 2007).

(I. H. Osman & Kelly, 1996) proponen la siguiente definición: “Los metaheurísticos son métodos aproximados diseñados para resolver problemas de optimización combinatoria, en los que los heurísticos clásicos no son efectivos”.

Una metaheurística puede tener cuatro componentes: el espacio inicial de soluciones, los motores de búsqueda, estrategias de aprendizaje y orientación y la gestión de estructuras de información (Ibrahim H Osman, 2001). En este sentido, y siguiendo la evolución de la literatura general del VRP, desde la última década del siglo XX, los enfoques metaheurísticos comenzaron a ser aplicados a la solución del VRP con flota heterogénea (Baldacci et al., 2007), dentro de los principales a portes al MFVRP se encuentran:

2.2.1 Búsqueda Secuencial por Entornos

Según Yepes (2002), el uso de operadores, que permiten el paso de una solución a otra de su entorno, mejora la función objetivo mientras no se alcance un óptimo local. La idea central de las metaheurísticas basadas en las búsquedas por entornos, se fundamenta en la degradación estratégica de las opciones que mejoran las de su vecindario para alcanzar un nuevo óptimo relativo.

A continuación se presentan dos metaheurísticas pertenecientes a este grupo, las cuales han sido empleadas para solucionar el MFVRP:

2.2.1.1 Algoritmo De Recocido Simulado

El recocido simulado (*SA*[23]) es una metaheurística capaz de evitar ser atrapado en un óptimo local mediante la aceptación, en pequeña probabilidad, por soluciones peores durante sus iteraciones (Bräysy et al., 2006). El procedimiento de optimización del *SA* alcanza un (cercano) mínimo global, para ello se parte de una solución inicial aleatoria. En cada iteración, el algoritmo toma una nueva solución del vecindario predefinido de la actual solución. El valor de la función objetivo de esta nueva solución es entonces comparado con el valor de la solución actual, con el fin de determinar si una mejora se ha conseguido. Si de hecho el valor de la función objetivo de la nueva solución es mejor, la nueva solución se convierte en la solución actual desde la cual la búsqueda continúa procediendo con una nueva iteración (Lin, Yu, & Chou, 2011).

Bräysy et al. (2006) propusieron una nueva metaheurística de recocido determinista para el FSMVRPTW. La metaheurística se basa en tres fases: (i) soluciones iniciales se generan por medio de una heurística basado en ahorros, combinando estrategias de diversificación con mecanismos de aprendizaje, (ii) se realiza un intento para reducir el número de rutas en la solución inicial con un nuevo procedimiento de búsqueda local y (iii) la solución de la segunda fase se mejora aún más mediante un conjunto de cuatro operadores de búsqueda local que están incrustados en un marco de recocido determinista para guiar el proceso de mejora.

2.2.1.2 Búsqueda Tabú

La Búsqueda Tabú (Tabu Search - *TS*) tiene sus antecedentes en métodos diseñados para cruzar cotas de factibilidad u optimalidad local tratadas como barreras en procedimientos clásicos, e imponer y eliminar cotas sistemáticamente para permitir la exploración de regiones no consideradas en otros casos. Una característica distintiva de este procedimiento es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas. TS es también responsable de enfatizar el uso de los diseños estructurados para explotar los patrones históricos de la búsqueda, de forma opuesta a los procesos que confían casi exclusivamente en la aleatorización (Glover & Melián, 2003a).

[23] Simulated Annealing

La Búsqueda Tabú y variantes de la misma se han empleado para resolver el MFVRP. Tres procedimientos basados en búsqueda tabú se presentan en Osman y Salhi (1996), Gendreau et al. (1999) y Wassan y Osman (2002).

Para el algoritmo de búsqueda Tabú de Osman y Salhi (1996), la solución inicial se obtiene por el algoritmo de (Salhi & Rand, 1987) y se mejoró mediante el mecanismo de intercambio λ de (I.H. Osman, 1993), las etapas de este algoritmo son: Una solución inicial, luego se emplea un mecanismo de generación de vecindarios, posteriormente se evalúa el costo de un movimiento, se utiliza una estructuras de datos para la lista de candidatos de movimientos, con ello se emplea una lista tabú, con su respectiva condición y tamaño, finalmente se usa un criterio de aspiración y un criterio de parada (Osman & Salhi, 1996).

(Gendreau et al., 1999), proponen para el MFVRP un algoritmo bastante complicado. En primer lugar, hace uso de GENIUS. En segundo lugar, incorpora varias estrategias, como la penalización de la función objetivo, una solución inicial, una estructura de vecindarios, un estado tabú y criterios de aspiración y por último una post-optimización y cambio de flota. En tercer lugar, el algoritmo es incrustado dentro de un Procedimiento de Memoria Adaptativo (*AMP*[24]), una técnica de búsqueda desarrollada por (Rochat & Taillard, 1995) en el contexto del VRP.

(Wassan & Osman, 2002), combinan varias estrategias eficaces para mejorar la calidad global del algoritmo. Su metaheurística requiere la definición de (i) una solución inicial S, (ii) un mecanismo de vecindario para generar el conjunto de soluciones vecinos $N(S)$; (iii) una estructura de gestión de datos para un almacenamiento y recuperación de la información generada eficiente y (iv) el conjunto de componentes para el algoritmo *TS*.

De manera general, los algoritmos que emplean una Búsqueda secuencial por entornos obtienen excelentes resultados cuando se emplean en problemas de optimización combinatoria, de hecho son los métodos actualmente más empleados debido a la calidad de sus respuestas (Vélez & Montoya, 2007). La principal desventaja que muestran estas metaheurísticas tiene que ver con el tiempo computacional empleado para obtener una respuesta, además su implementación no es sencilla, ya que estos algoritmos presentan estructuras de funcionamiento que deben ser determinadas de la mejor manera, siempre adaptándolo a la instancia a solucionar.

[24] Adaptive Memory Procedure

2.2.2 Algoritmos Evolutivos

Una población de soluciones puede evolucionar hacia individuos de mayor aptitud si existen mecanismos de reproducción, bien cruzando información, si existen varias soluciones que originan nuevas, o simplemente de reproducción, cuando el conjunto se somete a criterios de supervivencia por selección (Yepes, 2002). Dentro de estos algoritmos encontramos aquellos que presentan cruzamiento de información, como los Algoritmos Genéticos el cual ha sido empleado para solucionar variantes del MFVRP.

2.2.2.1 Algoritmos Genéticos

Los Algoritmos Genéticos (*GA*[25]) han sido inspirados en el mecanismo de selección natural introducido por Darwin, de manera general son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización (Yepes, 2002). En este sentido, se aplican algunos operadores a una población de soluciones del problema en cuestión, de tal manera que la nueva población es mejor en comparación con el anterior, de acuerdo con una función de criterio preestablecido. Este procedimiento se aplica para un número preseleccionado de iteraciones y la salida del algoritmo es la mejor solución encontrada en la última población o, en algunos casos, la mejor solución encontrada durante la evolución del algoritmo (Muhammad et al., 2011).

(Tan et al., 2006), propusieron un algoritmo evolutivo hibrido multi-objetivo (HMOEA[26]) para solucionar el Problema de Ruteo de Vehículos con Camiones y Remolques (TTVRP[27]), esta metaheurística incorpora una heurística de búsqueda local y el concepto de Optimización de Pareto. El HMOEA optimiza los objetivos del TTVRP al mismo tiempo, sin la necesidad de agregar múltiples criterios en función del compromiso.

Los algoritmos evolutivos al igual que los métodos de búsqueda secuencial por entornos generan excelentes resultados, pero también presentan las mismas desventajas ya que el tiempo computacional requerido para obtener una respuesta es relativamente grande y su implementación no es sencilla.

[25] Genetic Algorithms
[26] Hybrid Multiobjective Evolutionary Algorithm
[27] Truck and Trailer Vehicle Routing Problems

Finalmente existen otras heurísticas y metaheurísticas que se emplearon para solucionar el MFVRP, estos algoritmos fueron diseñados por sus autores tomando como base los métodos más conocidos, los cuales se presentaron en este capítulo. Si se quiere conocer un poco más sobre estos algoritmos deben referirse a los trabajos de (Renaud & Boctor, 2002); (Baldacci et al., 2007) y (Gendreau et al., 2007).

A manera de conclusión general se puede decir que aunque existen varios algoritmos para desarrollar el MFVRP, sólo unos pocos tienen la capacidad de generar respuestas de alta calidad, sin embargo su esfuerzo computacional es mucho mayor que los métodos más sencillos de efectuar. En este sentido, la decisión de ejecutar un método u otro recae en dos opciones: Escoger un algoritmo sencillo y rápido de implementar, aunque sus respuestas no sean de alta calidad o Escoger un algoritmo robusto que genere soluciones de alta calidad, aunque su desarrollo sea algo complejo de efectuar. Para este trabajo de grado se decidió optar por la segunda decisión ya que el objetivo último de la tesis es minimizar (en mayor medida) los costos asociados al transporte.

3. DESCRIPCIÓN DE LA METAHEURÍSTICA BÚSQUEDA TABÚ.

Desde que se creó en 1986, más de un centenar de publicaciones presentan las aplicaciones de la Búsqueda Tabú a los diversos problemas combinatorios, que han aparecido en la literatura de la investigación de operaciones (Gendreau, 2002). En varios casos, los métodos descritos proporcionan soluciones muy cerca de la optimalidad y se encuentra entre los algoritmos más efectivos, si no es el mejor, para hacer frente a los difíciles problemas encontrados en la vida real. La búsqueda Tabú surge, en un intento de dotar de "inteligencia" a los algoritmos de búsqueda local. Según Fred Glover, su primer definidor, "la búsqueda tabú guía un procedimiento de búsqueda local para explorar el espacio de soluciones más allá del óptimo local" (Glover, 1986).

Esta metaheurística toma de la Inteligencia Artificial el concepto de memoria y lo implementa mediante estructuras simples con el objetivo de dirigir la búsqueda teniendo en cuenta la historia de ésta, es decir, el procedimiento trata de extraer información de lo sucedido y actuar en consecuencia. En este sentido puede decirse que hay un cierto aprendizaje y que la búsqueda es inteligente (Martí, 2003). La búsqueda Tabú permite moverse a una solución aunque no sea tan buena como la actual, de modo que se pueda escapar de óptimos locales y continuar estratégicamente la búsqueda de soluciones aún mejores (Díaz et al., 1996).

3.1 Conceptos de la Búsqueda Tabú

En esta sección se describirán los principales conceptos de *TS,* ya que su extensión bibliográfica es bastante amplia, encontrándose libros y una gran cantidad de documentos sobre esta metodología. Como se expresó anteriormente la característica que distingue a *TS* es el uso de la memoria, la cual tiene una estructura basada en una lista tabú y unos mecanismos de selección del siguiente movimiento.

3.1.1 Lista Tabú

La lista tabú es una lista donde se registran aquellas soluciones o atributos de soluciones que no deben ser elegidas (Glover & Laguna, 2007). La lista tabú no es

una enumeración de soluciones, ya que almacenar soluciones completas, aún en pequeñas cantidades, y comparar cada solución con las que hay en la lista es muy costoso computacionalmente. Para acopiar información sobre el pasado reciente, la búsqueda tabú almacena los últimos movimientos, un movimiento es una operación por medio de la cual se alcanza una solución vecina a partir de la solución actual (Vélez & Montoya, 2007).

Una forma sencilla de construir una lista tabú consiste en que cada vez que se realiza un movimiento, se introduce su inverso (si se pasó de x_0 a x_1, el inverso es x_1 a x_0) en una lista circular, de forma que los elementos de dicha lista están penalizados durante un cierto tiempo. Por tanto, si un movimiento está en la lista tabú no será aceptado, aunque aparentemente sea mejor solución que la solución actual. Más adelante se explicará la manera de invalidar la clasificación Tabú por medio de los criterios de aspiración. La lista tabú puede contener; Soluciones visitadas recientemente, Movimientos realizados recientemente o Atributos o características que tenían las soluciones visitadas.

3.1.2 Tamaño de la Lista Tabú (*Tabu Tenure*)

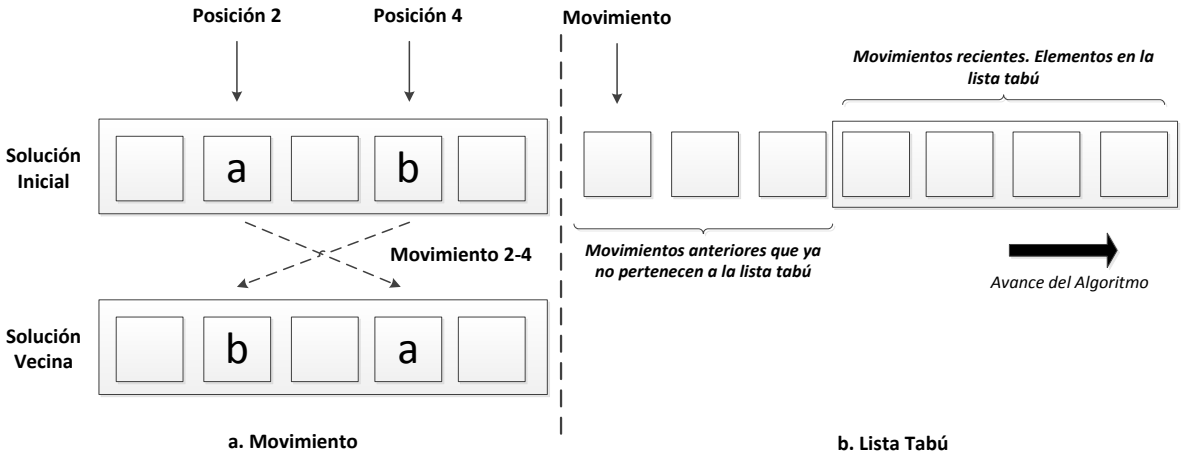
El tamaño de la lista tabú es el tiempo o número de iteraciones que un elemento (movimiento o atributo) permanece en la lista tabú, a esto también se le conoce como **Periodo Tabú** (Glover & Melián, 2003a).

Por ejemplo, si una solución se puede representar con un vector, como se ilustra en la **Figura 3.1**, un movimiento puede consistir en intercambiar la posición de dos elementos en el vector. Nótese que para almacenar el movimiento sólo es necesario almacenar dos valores (Ahuja, Magnati, & Orlin, 1988), (Ahuja et al., 1994) y que esta información por sí sola no representa una solución (**Figura 3.1a**). La lista tabú es una lista en la cual se almacenan los movimientos recientes por un lapso denominado tiempo de permanencia, como se muestra en la **Figura 3.1b**.

El período tabú puede ser distinto para diferentes tipos o combinaciones de atributos, y con un mayor nivel de desarrollo, pueden variar también sobre diferentes estados del proceso de búsqueda. Estas variaciones del período tabú de los atributos hace posible crear diferentes formas de balance entre las estrategias de memoria a corto y a largo plazo. Por lo tanto, para determinar cuándo son aplicables determinadas restricciones tabú, obtenidas a partir de los estados tabú de determinados atributos, es necesario disponer de funciones de

memoria que permitan almacenarlos eficaz y eficientemente (Glover & Melián, 2003a).

Figura 3.1 Estructuras básicas en la búsqueda tabú



Fuente: Adoptado de (Vélez & Montoya, 2007), pág. 107.

En este contexto y como se dijo anteriormente, es posible invalidar la clasificación tabú de un movimiento o atributo, esto por medio de un Criterio de Aspiración, en la siguiente sección se explicará brevemente este concepto.

3.1.3 Criterio de Aspiración

En ciertas ocasiones existen restricciones Tabú que son demasiado potentes: las cuales pueden prohibir movimientos atractivos, incluso cuando no hay peligro de ciclado, o pueden conducir a un estancamiento general del proceso de búsqueda. Por lo tanto, es necesario el uso de dispositivos algorítmicos que permitan revocar (cancelar) dichas restricciones tabú. A estos algoritmos se les llama Criterios de Aspiración, el más simple y más comúnmente utilizado consiste en permitir un movimiento, incluso si es tabú, si resulta en una solución con un valor objetivo mejor que la solución conocido recientemente (Glover & Melián, 2003a).

Las aspiraciones son de dos clases; **Aspiraciones de movimiento,** cuando se satisface esta aspiración, revoca la condición tabú del movimiento. **Aspiraciones de atributo, c**uando estas aspiraciones se satisfacen revocan el estado tabú del

atributo. En este último caso el movimiento puede o no cambiar su condición de tabú, dependiendo de si la restricción tabú puede activarse por más de un atributo. Si se desea conocer más sobre los criterios de aspiración, el lector debe referirse a (Glover & Melián, 2003a) y (Glover & Laguna, 2007).

3.2 Características de la Búsqueda Tabú

3.2.1 Uso de Memoria

Las estructuras de memoria de la búsqueda tabú funcionan mediante referencia a cuatro dimensiones principales, consistentes en la propiedad de ser Reciente, En Frecuencia, En Calidad y en Influencia (Glover & Melián, 2003b):

Las memorias basadas en lo **reciente** y en **frecuencia** se complementan la una a la otra para lograr el balance entre **intensificación** y **diversificación** que todo proceso de búsqueda heurística debe poseer.

La dimensión de **calidad** hace referencia a la habilidad para diferenciar la bondad de las soluciones visitadas a lo largo del proceso de búsqueda. De esta forma, la memoria puede ser utilizada para la identificación de elementos comunes a soluciones buenas o a ciertos caminos que conducen a ellas. La calidad constituye un fundamento para el aprendizaje basado en incentivos, donde se refuerzan las acciones que conducen a buenas soluciones y se penalizan aquellas que, por contra, conducen a soluciones pobres.

Por último, la cuarta dimensión de memoria, referida a la **influencia**, considera el impacto de las decisiones tomadas durante la búsqueda, no sólo en lo referente a la calidad de las soluciones, sino también en lo referente a la estructura de las mismas.

El uso de memoria en *TS* es tanto **explícito** como **implícito**. En el primer caso, se almacenan en memoria soluciones completas, generalmente soluciones élite visitadas durante la búsqueda, mientras que en el segundo caso, se almacena información sobre determinados atributos de las soluciones que cambian al pasar de una solución a otra. Estos dos tipos de memoria son complementarios, puesto que la memoria explícita permite expandir los entornos de búsqueda usados

durante un proceso de búsqueda local, mediante la inclusión de soluciones élite, la memoria implícita los reduce prohibiendo determinados movimientos.

Una distinción importante en *TS* es la de diferenciar la **memoria de corto plazo** y la de **largo plazo**, cada tipo de memoria tiene sus propias estrategias, sin embargo, el efecto de la utilización de ambos tipos de memoria es el mismo: *modificar el vecindario* $N(x)$ *de la solución actual* x *(convertirlo en un nuevo vecindario* $N^*(x)$ *)* (Martí, 2003).

3.2.1.1 Memoria de Corto Plazo

La memoria a corto plazo más utilizada generalmente en la literatura, almacena los atributos de las soluciones que han cambiado en el pasado reciente. Este tipo de memoria a corto plazo se denomina **memoria basada en lo reciente**. La forma más habitual de explotar este tipo de memoria es etiquetando los atributos seleccionados de soluciones visitadas recientemente como tabú-activos. Se considera que un atributo es tabú-activo cuando su atributo inverso asociado ha ocurrido dentro de un intervalo estipulado de lo **reciente**. Un atributo que no es tabú-activo se llama tabú-inactivo. De esta forma, aquellas soluciones que contengan atributos tabú-activos, o combinaciones particulares de los mismos, se convierten en soluciones tabú o prohibidas. La condición de ser tabú-activo o tabú-inactivo se llama el ***estado tabú*** de un atributo.

Aunque las restricciones tabú más comunes, cuyos atributos son los inversos de aquellos que definen las restricciones, tienen generalmente el objetivo de prevenir el ciclado, es necesario precisar que el objetivo final de *TS* no es evitar ciclos, pero a veces, un buen camino de búsqueda resultará en volver a visitar una solución encontrada anteriormente. El objetivo más general es continuar estimulando el descubrimiento de nuevas soluciones de alta calidad (Glover & Melián, 2003a).

3.2.1.2 Memoria de Largo Plazo

En algunas aplicaciones, los componentes de la memoria a corto plazo son suficientes para producir soluciones de alta calidad. Sin embargo, la inclusión de la memoria a largo plazo, así como de las estrategias asociadas a la misma, hacen del *TS* una estrategia más fuerte (Glover & Laguna, 2007).

La memoria de largo plazo o **memoria basada en frecuencia** proporciona un tipo de información que complementa la información proporcionada por la memoria basada en lo reciente, ampliando la base para seleccionar movimientos preferidos. Al igual que sucede en la memoria basada en lo reciente, la frecuencia a menudo está ponderada o descompuesta en subclases teniendo en cuenta las dimensiones de calidad de la solución e influencia del movimiento.

En esta estructura de memoria se registra la frecuencia de ocurrencias de los movimientos, las soluciones o sus atributos. Existen dos tipos de Frecuencia; de **transiciones,** es la cantidad de veces que una solución es la mejor o cantidad de veces que un atributo pertenece a una solución generada y la **Frecuencia de residencia,** que es la cantidad de iteraciones durante la cual un atributo pertenece a la solución generada.

3.2.2 Estrategias de Intensificación y Diversificación Simples

Las funciones de **intensificación** y **diversificación** en *TS* ya están implícitas en muchas de las prescripciones anteriores, pero se convierten especialmente relevantes en procesos de búsqueda de periodo largo (Glover & Laguna, 2007).

Las **estrategias de intensificación** crean soluciones agresivamente estimulando la incorporación de “atributos buenos”. En el periodo corto esto consiste en incorporar atributos que han recibido las mayores evaluaciones por los enfoques y criterios descritos anteriormente, mientras que en el intermedio a largo periodo consiste en incorporar atributos de soluciones de subconjuntos elite seleccionados. En otras palabras consiste en regresar a regiones ya exploradas para estudiarlas más a fondo. Un enfoque típico de la intensificación es reiniciar la búsqueda de la mejor solución actualmente conocida y "congelar" (punto de referencia) en el mismo los componentes que parecen más atractivas.

Por otro lado, las **estrategias de diversificación** generan soluciones que incorporan composiciones de atributos significativamente diferentes a los encontrados previamente durante la búsqueda, es decir, se visitan nuevas áreas no exploradas del espacio de soluciones. Para ello se modifican las reglas de elección para incorporar a las soluciones atributos que no han sido usados frecuentemente.

Una forma clásica de diversificación consiste en reiniciar periódicamente la búsqueda desde puntos elegidos aleatoriamente, si se tiene alguna información acerca de la región factible se puede hacer un "muestreo" para cubrir la región en lo posible; si no, cada vez se escoge aleatoriamente un punto de partida (método multi arranque). Otro método para diversificar propone registrar los atributos de los movimientos más utilizados en los anteriores movimientos, penalizándolos a través de una lista tabú a largo plazo y así explorar otros entornos.

Estos dos tipos de estrategias se contrapesan y refuerzan mutuamente de varias formas.

Los métodos de intensificación y diversificación que utilizan penalizaciones e incentivos representan sólo una clase de tales estrategias. Una colección mayor surge de la consideración directa de los objetivos de intensificación y diversificación, algunas de estas estrategias avanzadas son: Refuerzo por restricción, Re-encadenamiento de camino[28], Procedimientos de Listas de Candidatos, entre otros. Si se quiere conocer un poco más de cada uno de estos aspectos avanzados de Intensificación y Diversificación, se puede dirigir a (Glover & Melián, 2003b).

Como se mencionó antes, los algoritmos *TS* son hoy en día una de las mejores opciones para la mayoría de los problemas difíciles de optimización. Sin embargo, los tiempos de ejecución necesarios para detectar buenas soluciones suelen ser muy grandes en comparación con los métodos tradicionales, como las heurísticas constructivas, que requieren tiempos de ejecución relativamente cortos.

(Toth & Vigo, 2003), introducen un método de intensificación de búsqueda, perteneciente a la familia de las **Estrategias de Lista de Candidatos** llamada **Granular Tabu Search** (**GTS**) el método, explicado en detalle más adelante, se basa en el uso de vecindarios restringidos denominados *granular neighborhoods* (vecindarios granulares) obtenidos a partir de la eliminación de una gran cantidad de ejes de los vecindarios tradicionales que se consideren de poca probabilidad de pertenecer a buenas soluciones. Finalmente el tiempo necesario para alcanzar soluciones de alta calidad a menudo es mucho menor que el requerido cuando se utilizan los vecindarios completos correspondientes.

[28] PR, Path Relinking

3.2.3 Estrategia de Listas de Candidatos

Para problemas grandes, donde se puede tener muchos elementos, o para problemas donde estos elementos pueden ser costosos de examinar, la orientación de elección agresiva de *TS* hace altamente importante aislar un subconjunto candidato del entorno, y examinar este subconjunto en vez del entorno completo. En tales métodos, el subconjunto de movimientos se referencia mediante una lista que identifica sus elementos definitorios (tales como índices de variables, nodos y arcos), y por tanto estos métodos han adquirido el nombre de estrategias de listas de candidatos (Glover & Laguna, 2007).

La **Tabla 3.1** muestra los elementos de la memoria en *TS* mencionados anteriormente.

Tabla 3.1 Características de los tipos de memoria.

Memoria	Atributo	Estrategia	Ámbito
Corto Plazo	Reciente	Tabú - Aspiración Lista de Candidatos	Local
Largo Plazo	Frecuente	Intensificación - Diversificación	Global

Fuente: Adoptada de Martí (2003), Pág. 40.

3.3 Búsqueda Tabú Granular (GTS)

El GTS es una variante relativamente nueva, del bien conocido enfoque de Búsqueda Tabú. El método utiliza una herramienta de intensificación-diversificación eficaz que puede ser aplicado con éxito a una amplia clase de problemas de teoría de gráficos y optimización combinatoria. El GTS se basa en el uso de vecindarios restringidos drásticamente, que no contengan movimientos que involucran elementos que no son susceptibles de pertenecer a buenas soluciones factibles. Estos vecindarios restringidos se llaman **granulares**, y pueden ser vistos como una aplicación eficiente de las estrategias de lista de candidatos propuesta para los algoritmos de búsqueda tabú (Toth & Vigo, 2003).

Dentro de los parámetros generales de TS, los vecindarios granulares pueden ser vistos como una implementación de estrategias de listas de candidatos (Glover & Laguna, 2007). De hecho, la búsqueda se limita a un conjunto élite de soluciones

de vecindario, y el criterio usado para seleccionarlas se fija de antemano. Ideas similares han sido utilizadas por varios autores para acelerar algoritmos de búsqueda local. Por ejemplo, se usa la lista de candidatos dentro de algoritmos 2-opt y 3-opt para el problema del viajante de comercio (Johnson & Mcgeoch, 1997).

3.3.1 Vecindarios Granulares

Los vecindarios granulares se obtienen eliminando movimientos que implican sólo elementos que no son susceptibles de pertenecer a soluciones factibles de alta calidad (Berbotto, Garc, & Nogales, 2013). Por lo tanto, estos vecindarios conducen a búsquedas menos miopes en los cuales sólo los movimientos potencialmente prometedores son en realidad evaluados en cada iteración.

En este sentido, los vecindarios conducen a una drástica reducción en el tiempo computacional que requiere cada iteración de TS, por lo cual pueden ser examinados en considerablemente menos tiempo que los completos. En cada iteración es posible buscar un vecindario múltiple mucho más grande, que se define como la unión de diferentes vecindarios granulares. De esta manera el tiempo necesario para alcanzar soluciones de alta calidad a menudo es mucho menor que el requerido cuando se utilizan los vecindarios completos correspondientes (Toth & Vigo, 2003).

3.4 Metodología de la Búsqueda Tabú

La búsqueda tabú en su versión básica procede como cualquier algoritmo de búsqueda: Dada una solución x se define un entorno o vecindario $N(x)$, se evalúa y se “mueve” a una mejor solución pero, en lugar de considerar todo el entorno o vecindario la búsqueda tabú define el entorno reducido $N^*(x) \therefore N^*(x) \subseteq N(x)$ como aquellas soluciones disponibles (no tabú) del entorno de x. Así, se considera que a partir de x, sólo las soluciones del entorno reducido son alcanzables (Martí, 2003). La idea básica de la Búsqueda Tabú se presenta a continuación (Sait & Youssef, 1999)**.**

Algoritmo 3.1
Sea $x^* \in \Omega$ la mejor solución encontrada.
Sea N_A el nivel de aspiración.

Hacer $x^* \leftarrow x_0$
Mientras no se cumpla el criterio de parada:
    Seleccionar al azar el conjunto de vecinos $V \subseteq N(x^*)$ de vecinos x^*
    Sea $x' \in V$ la mejor solución V
    Si el movimiento que generó x' no está en la lista Tabú
        Aceptar el movimiento haciendo $x^* \leftarrow x'$
        Actualizar la lista Tabú incorporando a la lista el movimiento que generó x'
        Actualizar N_A
    Sino
        Si se cumple el criterio de aspiración
            Aceptar el movimiento haciendo $x^* \leftarrow x'$
            Actualizar la lista Tabú incorporando a la lista el movimiento que generó x'
        Fin
    Fin
Fin

En la **Tabla 3.2** se presentan los componentes del algoritmo de Búsqueda Tabú que deben estar determinados con claridad y precisión.

Tabla 3.2 Componentes del Algoritmo de Búsqueda Tabú.

Componente	Símbolo	Descripción
Función Objetivo	$f(x)$	La función que nos permite identificar la calidad de la solución, en este caso minimizar la distancia total recorrida.
Espacio de Soluciones	S	El conjunto de soluciones posibles R^n
Vecindario	$N(x)$	Las posibles soluciones que pueden seguir a x. Estos vecinarios estan restringidos a un grupo selecto de posibles soluciones, en el caso de los **vecindarios granulares** $N(x) \subset S$
Lista Tabú	$T(x,k)$	Lista de las posibles soluciones que no pueden ser elegidas en la actual iteración. Son los "L" últimos movimientos.
Lista de Candidatos	$A(x,k)$	Soluciones que tienen algún atributo que las hace elegibles aún si estuvieran en la lista tabú. En el caso del GTS, los candidatos deben cumplir con cierta condición para aspirar a ser solución en la iteración k.
Número máximo de iteraciones	$Maxiter$	Para evitar que el algoritmo entre en un proceso sin fin, es aconsejable establecer una cota superior de iteraciones posibles, este parámetro se define de antemano.

Fuente: Adaptado de (Riojas Cañari, 2005), pág. 49.

4. CASO ESTUDIO

La compañía seleccionada para aplicar el algoritmo, es una empresa del sector industrial de la ciudad de Cali, la cual se dedica a la elaboración de grasa alimenticia, esta empresa cuenta con 79 clientes en toda la ciudad. A continuación se describirá brevemente el desarrollo del caso estudio, lo cual se hará en diferentes etapas, cubriendo así el segundo objetivo de este trabajo de grado.

4.1 Identificación del Sistema

El primer paso para la consecución de los objetivos de este proyecto es conocer los antecedentes del caso estudio. Los datos fueron suministrados por la empresa, la cual realiza despachos en toda la zona urbana de la ciudad; en las tablas proporcionadas por la compañía se detalla la demanda por semana, se analizan los datos, obteniendo una totalidad de 79 clientes, los cuales una vez identificados se procede a realizar el levantamiento de la información.

4.1.1 Supuestos y Limitaciones

A continuación se mostrarán los supuestos que se deben tomar en cuenta para la realización del caso estudio:

- ✓ Los pedidos de acuerdo a los requerimientos de cada cliente, serán entregados completos. Se asume que no existen pérdidas en el traslado de las mismas. Además los clientes no tendrán horarios de atención, es decir, el problema no considera ventanas de tiempo.
- ✓ La empresa cuenta con 8 vehículos para el transporte de la mercancía, además los camiones pueden o no llenarse a la totalidad de sus capacidades de transporte. En el momento del transporte de la mercancía, los vehículos estarán sujetos a una velocidad constante de 45 Km/h, asumiendo que no existen problemas desde el punto de origen a su destino final.
- ✓ La flota de camiones es heterogénea por lo tanto los vehículos poseen diferentes costos y capacidades, así no se caracterice detalladamente cada costo.
- ✓ Se asumirá un tiempo de parada y un tiempo administrativo por cliente de 0,5h y 0,25h respectivamente.

4.1.2 Obtención de los datos

Una vez que se identificó el número de clientes se prosiguió con la obtención de los datos, para esto se hizo uso de la herramienta ofimática, Excel, en el cual se emplea una hoja de cálculo para generar una matriz donde se insertan los antecedentes de demandas por los clientes. En esta misma hoja se muestran además las coordenadas correspondientes para cada cliente en la ciudad, dichas coordenadas se encuentran en metros (por confidencialidad de la empresa, no se pueden mostrar las direcciones de los clientes).

Tabla 4.1 Recopilación de datos.

Cliente	Coordena X (m)	Coordenada Y (m)	Demanda (Kg)	Angulo (Rad)
Depósito	0	0		
Cliente 1	3.503,75	2.033,35	1.654,20	0,5258347
Cliente 2	3.938,93	1.725,31	355,03	0,4128425
Cliente 3	5.241,98	2.009,45	5.032,80	0,3660601
Cliente 4	3.760,75	3.964,64	4.660,00	0,8117831
Cliente 5	3.099,29	5.080,71	750,00	1,0230473

Fuente: Adaptación de los datos suministrados por la empresa y tabulados por el autor.

4.1.3 Flota de Camiones

Para este caso estudio se utilizará una flota heterogénea de camiones; 1 camión de 18.000 Kg, 1 camión de 15.000 Kg, 4 camiones de 12.000 Kg y 2 camiones de 8.000 Kg, los cuales estarán disponibles para todos los conjuntos de rutas generados, siempre y cuanto se respete la restricción de capacidad de cada vehículo. Los camiones empiezan su recorrido en el Depósito que estará ubicado en la coordenada (0,0), y realizará su tour visitando la cantidad de clientes asignados a la ruta, para luego regresar a su punto de partida.

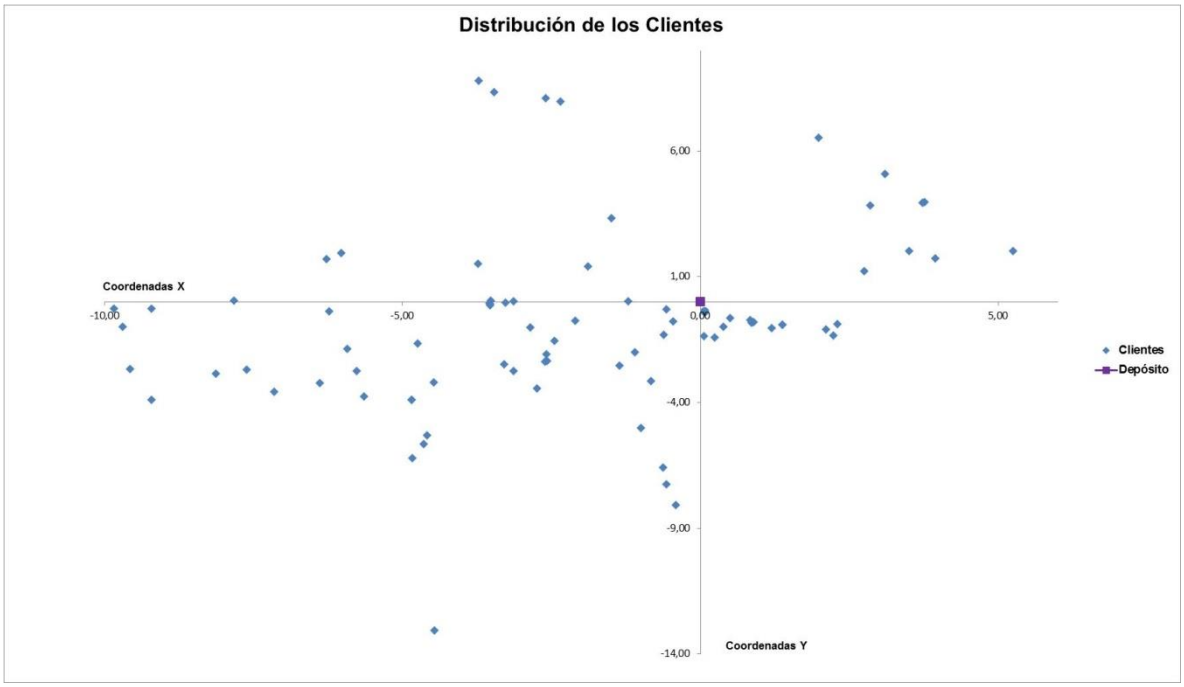
En este punto es necesario aclarar que el algoritmo fue diseñado para no limitarse por la cantidad total de camiones fijados de antemano, es decir, si el algoritmo ve la necesidad que “crear” otro camión de 8.000 Kg lo hará, aún si ya se han empleado los 2 vehículos disponibles que existían al principio. Este detalle se discutirá con más cuidado más adelante.

4.1.4 Distribución de los Clientes

Como se dijo anteriormente la empresa cuenta con 79 clientes distribuidos en la ciudad de Cali, a estos clientes se les realiza su respectiva ubicación en coordenadas cartesianas con el fin de resolver el modelo en función de las distancias entre cada punto. En este sentido el algoritmo está diseñado para encontrar la mínima distancia total recorrida del sistema (función objetivo), lo que conlleva a una disminución de los costos de transporte.

Para un mejor acercamiento en la solución del modelo, se opta por ubicar el centro de distribución en la coordenada (0,0), de tal manera que se reorganicen también las coordenadas de los clientes en función del depósito. En la **figura 4.1** se muestra la distribución de los clientes, donde el punto de color violeta representa el CD.

Figura 4.1 Ubicación de las Coordenadas de los Clientes.



Fuente: Adoptado de los datos suministrados por la empresa y elaborado por el autor.

5. PROPUESTA DE BÚSQUEDA TABÚ PARA LA RESOLUCIÓN DEL MODELO MFVRP.

En esta sección se mostrará en forma detallada el algoritmo implementado, incluyendo la generación de la solución inicial y la forma en la que se crean y se modifican las rutas de la ejecución. Además se explicará la forma en que se generan los vecindarios mediante el movimiento definido por la operación que se desarrolló. Al terminar este capítulo se habrá cubierto lo expresado en el tercer objetivo de esta tesis.

5.1 Descripción General

Algoritmo 5.1

Leer la instancia a ejecutar
Definir criterio de parada
Establecer el Periodo Tabú
Identificar y ordenar camiones
Definir los parámetros de ejecución del GTS
Generar la solución inicial S_0
Generar documento y gráfico de salida
Fijar $S^* = S_0$
Iniciar la lista Tabú
Mientras no se cumpla el criterio de parada haga
    Crea lista de Candidatos según parámetro de GTS
    Analiza posibles movimientos dentro de rutas en todas las rutas
        Si es Candidato y no está en la lista tabú entonces
            Calcula costo la insertar el nuevo cliente en ruta y crea ruta
            Calcula la factibilidad de la inserción en cuanto a capacidad del vehículo
            Si es necesario se penaliza $ExtraCap * PenCap$
        Fin
        Si $S^* + Insertion + ExtraCap * PenCap \leq S_M$ entonces
            $S_M = S^* + Insertion + ExtraCap * PenCap$
        Fin
    Se guardan los movimientos y se agregan a la lista de candidatos
    Se agregan a la lista tabú
    Se construyen las nuevas rutas con camiones
    Descontar el Periodo Tabú de los movimientos presentes en la lista tabú
    Se agrega al documento de salida la solución del GTS y se crea grafico de salida
Fin

El **algoritmo 5.1** muestra de manera muy general el funcionamiento del método tabú implementado. Al comenzar la ejecución el programa lee el caso estudio a ejecutar, se definen los criterios de parada y el periodo tabú, se identifican y ordenan los camiones a emplear y se definen los parámetros del GTS.

El primer paso del algoritmo consiste en generar la solución inicial S_0 que sirve de base en la búsqueda de soluciones. Para este caso estudio se implementó el método de Barrido para la solución inicial, el cual será explicado en detalle en las próximas secciones. La solución hallada se almacena inicialmente en S^* que representa la mejor solución encontrada hasta el momento. Al finalizar esta parte se genera un documento de salida con la solución inicial, además se crea un gráfico donde se muestra cada ruta y la secuenciación de la misma.

Luego comienza el proceso iterativo basado en TS que recorre el espacio de soluciones intentando en cada paso mejorar la solución hallada. Como se dijo anteriormente se implementará una estrategia de lista de candidatos, con el fin de reducir el número de movimientos poco prometedores, los mejores movimientos forman parte de un vecindario, llamados Vecindarios Granulares, los cuales dependen del parámetro de dispersión $\boldsymbol{\beta}$ (Toth & Vigo, 2003). Este ajuste se aplica sobre el grafo de vecindad y el resultado impacta directamente en el método de generación de vecindarios que será explicado más adelante.

A continuación empieza el proceso de generación y evaluación de vecindarios. Si los movimientos están dentro de la lista de candidatos y no forman parte de la lista tabú, se realiza la operación de Inserción, con esto el algoritmo genera un vecindario de la solución actual aplicando cada combinación posible de los movimientos definidos por la operación de inserción. El vecindario resultante contiene sólo los movimientos válidos que son vecinos a la solución actual ya que los movimientos inválidos son descartados en el momento de la generación. Una vez que sea viable la inserción se construye la nueva ruta. La ruta anterior puede que sea viable desde el punto de vista del costo de la inserción, pero puede ser infactible en cuanto a la capacidad del vehículo, por eso es necesario calcular dicha factibilidad, si el nuevo movimiento excede la capacidad del camión se penaliza con un valor, con el fin de que sea menos atractivo y que afecte negativamente a la función objetivo. Esta parte del algoritmo permite explorar soluciones malas y moverse hacia otras soluciones con el fin de encontrar un óptimo global y no sólo un óptimo local.

Una vez finalizada la generación de los vecindarios, se procede a identificar la nueva solución S_M que reemplazará la S^* en el procedimiento de búsqueda. Para esto se determina si la suma de la S^* con el costo de inserción y la capacidad extra penalizada, es menor que la solución mejor S_M. Si se cumple la condición anterior se actualiza S_M como la suma de las tres variables anteriores.

Una vez elegido la nueva solución se procede a guardar los movimientos y se almacenan en la lista de candidatos. Posteriormente se marca como tabú por un periodo de iteraciones al movimiento inverso del utilizado para generar dicha solución, evitándose que se vuelva a la solución anterior. Paso siguiente se construyen las nuevas rutas junto con los camiones respectivos. Por último, se descuenta en uno el valor del periodo tabú de los movimientos restantes presentes en la lista tabú y se modifica el documento de salida, al cual se le agrega la solución final encontrada con el GTS, además se crea un gráfico donde se muestra dicha solución.

Este procedimiento iterativo se realiza hasta que se cumpla el criterio de parada, es decir, hasta que se alcance la cantidad máxima de iteraciones.

5.2 Solución Inicial

El algoritmo seleccionado para la solución inicial fue el método de barrido, ya que es uno de los algoritmos más sencillo para la formación de rutas para un problema de enrutamiento, además es de fácil implementación y es capaz de generar buenas soluciones en poco tiempo. De manera general el algoritmo de barrido implementado se muestra en el algoritmo 5.2.

La solución inicial consiste en generar rutas válidas que satisfagan las demandas de todos los clientes. Se inicia con la construcción de la matriz de distancias mediante la ecuación 5.1 implementada en C++:

$$1.22 \times \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (5.1)$$

Las distancias encontradas son, trayectos rectos que no siguen el camino de las calles de la ciudad, lo cual conlleva a que el recorrido represente una mayor distancia. Según investigaciones de (Aponza & Rodriguez, 2006), el factor de conversión para Cali es de 1,22 el cual se obtuvo de la regresión lineal de las distancias calculadas con el software GoogleTM Earth y las reales obtenidas con GPS, por tal motivo la ecuación 5.1 tiene multiplicado dicho factor de conversión.

Algoritmo 5.2

```
Se construye la matriz de distancias
Calcular ángulos de los clientes y se ordenan crecientemente
Se identifican los camiones y ordenan decrecientemente
Mientras haya demanda por atender haga
    Mientras haya disponibilidad de camión haga
        Si CapActual + DemandaCliente < CapCamion entonces
            Asigne cliente al camión
            CapActual = CapActual + DemandaCliente
            Actualice demanda y longitud de ruta
        Si no
            Cambie de camión
            Si CapActual + DemandaCliente < CapCamion entonces
                Asigne cliente al camión
                CapActual = CapActual + DemandaCliente
                Actualice demanda y longitud de ruta
            Fin
        Fin
    Fin
    Crea otro camión
    Asigna nuevo camión a lista de camiones
Fin
Secuenciar rutas
Reportar rutas
Fin
```

Posteriormente se calculan los ángulos de cada uno de los clientes y se ordenan crecientemente tal y como dice el seudocódigo anterior, se identifica a cada camión y se ordenan decrecientemente, esto con el fin, de que se empiecen a utilizar los vehículos de mayor capacidad, a continuación se asignan los clientes ordenados a cada camión, siempre respetando la restricción de capacidad. Si todavía hay demanda por atender, pero no existe un camión disponible para cubrir dicha demanda, el algoritmo crea otro camión con capacidad mínima para suplir la demanda, así todos los camiones enunciados en el problema ya hayan sido empleados, la característica de este vehículo es que es el de menor capacidad que pueda atender al cliente. El procedimiento del barrido se replica para las 79 posibilidades de inicio, es decir, una vez ordenados los clientes con respecto a sus ángulos, se inicia la asignación de clientes a los camiones empezando con el segundo cliente. Finalmente se escoge la solución que haya arrojado el mejor resultado.

Una vez se hayan creado todas las rutas, se empieza con la secuenciación de las mismas, para ello se empleó el algoritmo del vecino más cercano (KNN), ya que es uno de los métodos más sencillos y directos para resolver un TSP, su desarrollo comienza tomando el nodo más cercano al último nodo de la ruta que se haya ruteado y lo suma al final de la misma, y así sucesivamente hasta terminar los nodos de la red (Marti, 2004). El algoritmo genera una solución inicial válida donde la cantidad de vehículos puede exceder la cantidad disponible enunciada en el problema. La ejecución futura de la metaheurística disminuirá eventualmente la cantidad de vehículos en la búsqueda de soluciones de menor costo.

Al finalizar todo el procedimiento se reporta cada ruta creada con la demanda atendida, la distancia recorrida y el camión que atendió dicha demanda, además se reporta el valor de la función objetivo, la cual será empleada como solución inicial en la fase del GTS.

5.3 Solución de la Búsqueda Tabú Granular – GTS

A continuación se describirá como el algoritmo crea un conjunto de los mejores movimientos, los cuales serán evaluados para que minimicen la distancia total recorrida, además se describirá como se construyen y modifican las rutas.

5.3.1 Generación de Vecindarios

Como es generalmente el caso, el VRP es definido en una gráfica completa, $G = (V, A)$ donde V es el conjunto de vértices y A es el conjunto de arcos. En dicha gráfica se puede observar que los arcos "largos" (es decir, de alto costo) tienen una pequeña probabilidad de ser parte de las soluciones de alta calidad, tal y como lo demuestran (Toth & Vigo, 2003) en su documento. Ellos concluyen, que una posible manera de acelerar la búsqueda de un vecindario es, limitar lo más posible la evaluación de los movimientos, analizando sólo movimientos élite.

Para perseguir esta idea se crea un nuevo gráfico disperso $G' = (V, A')$. Este gráfico incluye todos los arcos que deben ser considerados para su inclusión en la solución actual: todos los arcos "cortos" y un subconjunto relevante de otros arcos importantes, tales como los que inciden al depósito y los que pertenecen a las mejores soluciones encontradas hasta ahora. La búsqueda de un vecindario

granular considera sólo los movimientos que se generan por los arcos pertenecientes a gráfico G', es decir, los movimientos que implican al menos un "buen" arco. Se debe tener en cuenta que lo anterior no significa que los arcos "largos" no se insertan en la solución actual, sino que son insertados a pesar de su poca probabilidad de formar parte de una buena solución.

En este caso estudio se ha implementado la regla de filtrado para definir arcos “cortos” utilizada por Toht y Vigo, 2003. Un arco es corto, si su costo no es mayor que el valor **umbral de granularidad ϑ**:

$$\vartheta = \beta . \frac{z'}{(n+k)'} \qquad (5.2)$$

Donde β es un parámetro de dispersión positivo adecuado, z' es el valor de la solución del método de Barrido y $(n+k)'$ es el número de clientes. Para este caso estudio el valor de β se determinó de manera experimental modificándolo en un rango de 0,5 a 5,0 tal y como se expresa en la bibliografía. El valor de β que arrojó los mejores resultados fue **4.5.**

En el algoritmo implementado cuando se evalúa un arco, y este es menor que el umbral de granularidad, se guarda en una matriz llamada candidatos “small”, por el contrario cuando no cumple con esta regla, el arco se almacena en una matriz denominada candidatos “big”. Posteriormente en la fase de búsqueda de los mejores movimientos se inicia con los candidatos “small”, si luego de algún tiempo no se obtienen buenos resultados, se realiza la búsqueda con los candidatos “big”.

5.3.2 Construcción y Modificación de rutas

La fase de la Búsqueda Tabú Granular empieza con los datos de la solución inicial, es decir, se guarda la secuencia de la ruta, la información de las rutas (demandas, distancias recorridas y camión por ruta) y el valor de la función objetivo. Posteriormente se crea la lista de candidatos, esta lista puede emplear dos tipos de candidatos; candidatos “small” y candidatos “big”, explicadas anteriormente. En la lista de candidatos se guardan los movimientos de la solución inicial, los arcos “cortos” y los arcos incidentes en el depósito. Posteriormente el algoritmo recorre todas las rutas y dentro de las rutas, con el fin de encontrar buenos movimientos, es decir, movimientos que estén dentro de la lista de candidatos y además que no estén dentro de la lista tabú. Cuando se encuentran

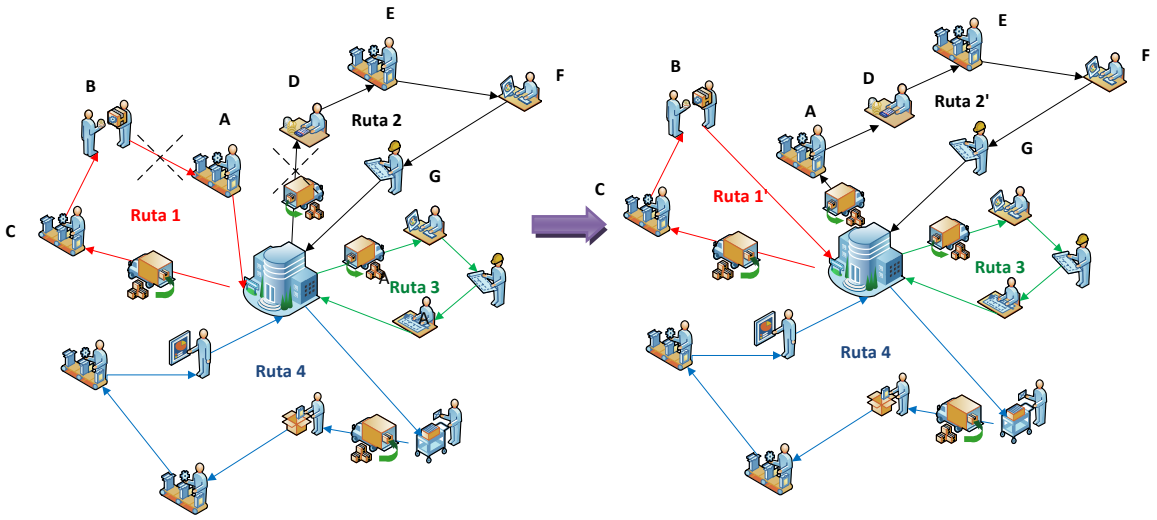
estos aspirantes, el algoritmo realiza la operación de inserción, tal y como se muestra en la **figura 5.1**.

Supongamos que se encontró un posible cambio (figura 5.1) el cual se va a evaluar, para ello se rompen el arco AB en la ruta 1 y el arco WD en la ruta 2, posteriormente se inserta el cliente ubicado en el vértice A de la ruta 1 a la ruta 2. Con esto la distancia total recorrida sufre la siguiente modificación:

$$Insertion = \big(d(WB) + d(AD)\big) - (d(AB) + d(WD))$$

Es decir, se suman las nuevas distancias al ubicar el cliente en la ruta 2, pero se restan las distancias de quitar el cliente de la ruta 1. Es necesario recordar que esta operación se efectúa si y sólo si los todos arcos cumplen con la regla de filtrado. El valor de la inserción conocido como delta Δ de ahora en adelante, pudo haber disminuido o no la distancia total recorrida, si hubo una mejora el movimiento se acepta temporalmente, en caso contrario se descarta y se vuelve a evaluar el siguiente movimiento.

Figura 5.1 Operación de inserción.



A) Identificación de arcos

B) Modificación de ruta

Fuente: Elaboración propia.

Una vez que el movimiento haya sido aceptado, se analiza la factibilidad en cuanto a la capacidad, ya que se ha agregado un nuevo cliente a la ruta 2, en este sentido se determina si la demanda de la ruta 2 excede la capacidad del camión que la atiende, en caso de que la exceda, el valor adicional de demanda se penaliza con un valor de 10, esto con el fin de que la función objetivo final se vea

afectada por un costo relacionado con la capacidad extra. Esta penalización permite explorar malas soluciones con el fin de salir de óptimos locales y dirigirse hacia mejores soluciones.

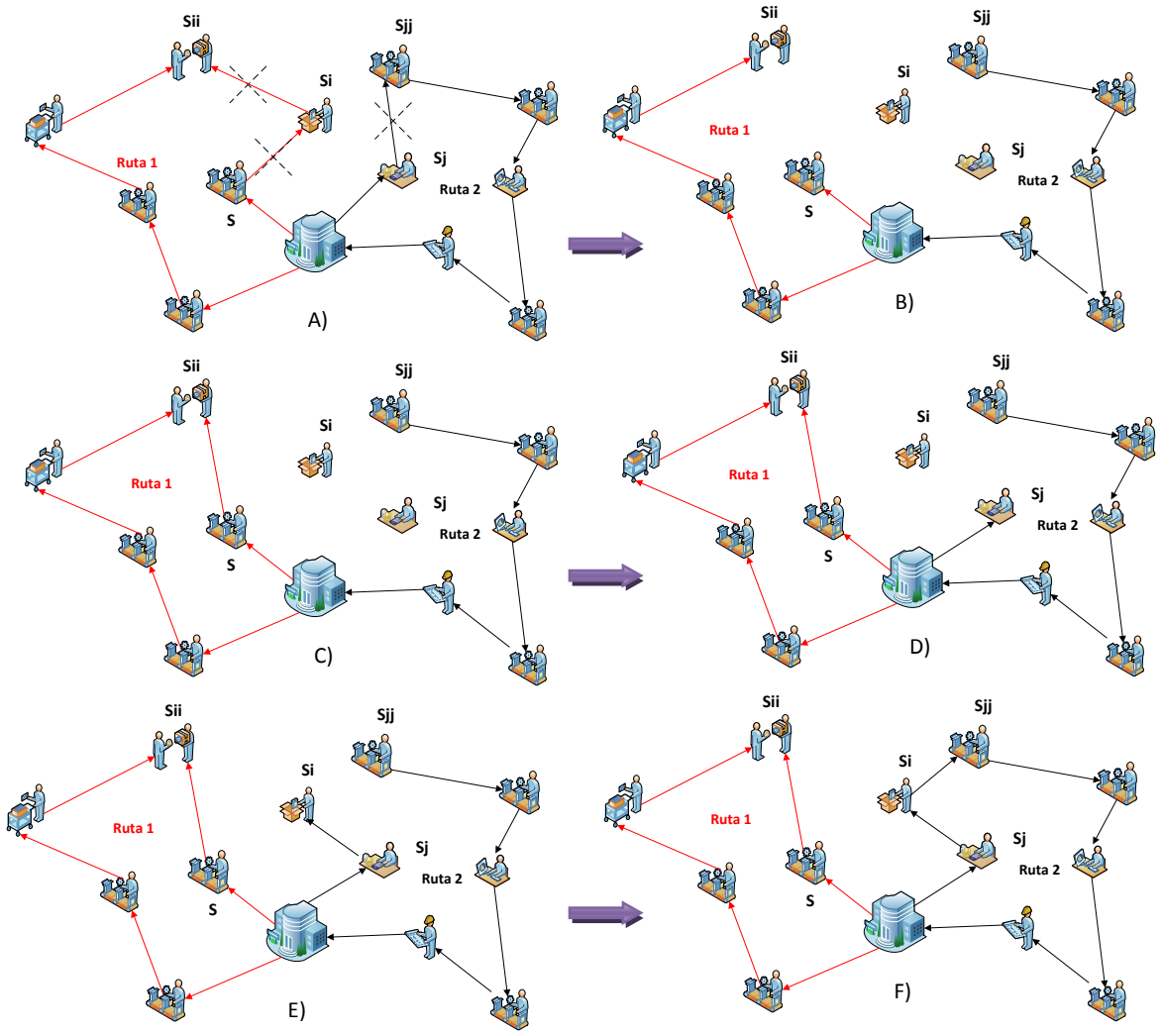
A continuación se determina el cambio o no de la función objetivo final, mediante la siguiente expresión:

Si $S^* + Insertion + ExtraCap \times PenCap \leq S_M$ entonces
$S_M = S^* + Insertion + ExtraCap * PenCap$
Fin

Donde $\boldsymbol{S^*}$ empieza como la solución inicial del método de barrido, pero posteriormente se va actualizando dependiendo de las distancias totales las cuales se hayan modificado; $\boldsymbol{Insertion}$ es la variación en las distancias cuando se evalúan movimientos candidatos mediante la operación de inserción; $\boldsymbol{ExtraCap \times PenCap}$ es el excedente de capacidad multiplicado con el valor de penalización y $\boldsymbol{S_M}$ es la mejor función objetivo, es decir, la función objetivo final generada en todo el proceso del GTS. Como se muestra en la expresión anterior, sólo se actualiza la función objetivo final cuando se logre minimizar o igualar dicho valor. Posteriormente los movimientos realizados se agregan en la lista tabú y se almacenan en la lista de candidatos. Luego de este paso se procede con la construcción de las rutas empleando los movimientos guardados en la fase anterior, este proceso es similar al mostrado en la **figura 5.2.**

El proceso de creación de las dos rutas modificadas es similar al proceso de inserción de un nuevo cliente, sólo que esta vez se emplearán los movimientos que mejoraron la función objetivo final. Primero el algoritmo determina que rutas se van a alterar, en este caso la ruta 1 y ruta 2 (**figura 5.2 a**), posteriormente desde el depósito suma el arco que va hasta S, agrega la demanda y la longitud recorrida a dicho punto, una vez hecho esto se guarda en la ruta 1 el vértice S (**figura 5.2 b**). Luego se realiza el mismo proceso anterior, sólo que se empieza desde S y se va hasta Sii (**figura 5.2 c**), como no hay más vértices que agregar en la primera ruta, se guarda la información de la ruta 1, es decir, el total de la demanda, la distancia total recorrida por el vehículo y los clientes de la ruta. En la ruta 2 se hace el mismo proceso, se adiciona el arco, se agrega la demanda y la longitud hasta el nuevo vértice agregado y se almacena el cliente a la ruta, del depósito a Sj luego de Sj a Si y luego de Si a Sjj, **figuras 5.2d, e y f** respectivamente.

Figura 5.2 Construcción de rutas 1 y 2 paso a paso.



Fuente: Elaboración propia.

Al finalizar, a las longitudes de las rutas no alteradas se le adiciona la distancia total recorrida en las dos rutas modificadas, guardando el resultado en $\boldsymbol{S}^*$, el cual se emplea posteriormente para modificar la función objetivo final $\boldsymbol{S_M}$, tal y como se explicó anteriormente. El último proceso que realiza el GTS es eliminar las rutas compuestas por un cliente, es decir, depósito – cliente – depósito, estas rutas son desarmadas y el cliente se ubica en la mejor posición de otra ruta. Luego de todo el proceso del GTS se descuenta el periodo tabú y se empieza un nuevo ciclo. Al completar el criterio de parada, se modifica el documento de salida, agregando las rutas, la información y el valor de la función objetivo generada por el GTS.

5.4 Detalles de la implementación

El implementar un algoritmo TS requiere determinar y diseñar sus características particulares y ajustar sus parámetros, tal como lo dice (Laguna, 1994). En esta subsección se explicará los detalles de la ejecución del algoritmo en general.

5.4.1 Periodo Tabú (Tabú Tenure)

Un parámetro de vital importancia en la búsqueda tabú es el periodo tabú. Algunas implementaciones utilizan un valor de periodo fijo a lo largo de la ejecución del algoritmo, mientras que otros autores realizan adaptaciones dinámicas o selecciones al azar en un rango de $[minTenure, maxTenure]$. En este caso se utilizó un valor fijo de 7 determinado empíricamente dentro de un rango de $t_{min} = 5$ y $t_{max} = 10$ como en (Toth & Vigo, 2003).

5.4.2 Estructura de GTS

Una de las decisiones más importantes a la hora de implementar un algoritmo TS es la elección de la estructura Tabú que será utilizada para llevar a cabo el historial de la exploración. Como se dijo antes, en la práctica es imposible almacenar el historial completo de soluciones visitadas. Por este motivo se optó por almacenar atributos que se definen en la operación de inserción. Por ejemplo si se mueve el cliente Si de la ruta 1 a la 2, este movimiento se convertirá en tabú por un número de iteraciones y sólo saldrá de esta lista cuando haya pasado dicho periodo.

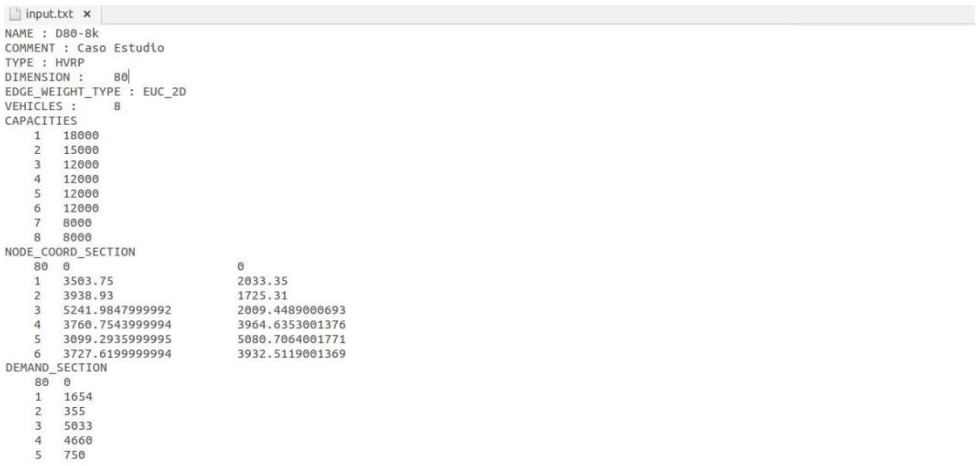
5.4.3 Archivo de Entrada

Al comenzar, el programa lee el archivo input.txt que contiene las datos del caso estudio. En la **figura 5.3** se muestra la una captura de pantalla del documento. El formato del archivo es el siguiente:

- ✓ **Identificación del caso estudio:** Es la información para saber qué tipo de VRP se está trabajando, la cantidad total de camiones y el número de nodos.

- ✓ **Capacities:** Se especifica uno a uno la capacidad de cada vehículo, desde el camión uno hasta el octavo.
- ✓ **Node_Coord_Section:** Es la sección donde se muestran las coordenadas X y Y de los clientes.
- ✓ **Demand_Section:** Contiene las demandas de cada cliente.

Figura 5.3 Captura de pantalla del archivo Input.txt.



input.txt

```
NAME : D80-8k
COMMENT : Caso Estudio
TYPE : HVRP
DIMENSION :    80
EDGE_WEIGHT_TYPE : EUC_2D
VEHICLES :     8
CAPACITIES
    1   18000
    2   15000
    3   12000
    4   12000
    5   12000
    6   12000
    7   8000
    8   8000
NODE_COORD_SECTION
    80  0                   0
    1   3503.75             2033.35
    2   3938.93             1725.31
    3   5241.9847999992     2009.4489000693
    4   3760.7543999994     3964.6353001376
    5   3099.2935999995     5080.7064001771
    6   3727.6199999994     3932.5119001369
DEMAND_SECTION
    80  0
    1   1654
    2   355
    3   5033
    4   4660
    5   750
```

Fuente: Elaboración propia.

5.4.4 Lenguaje de Programación

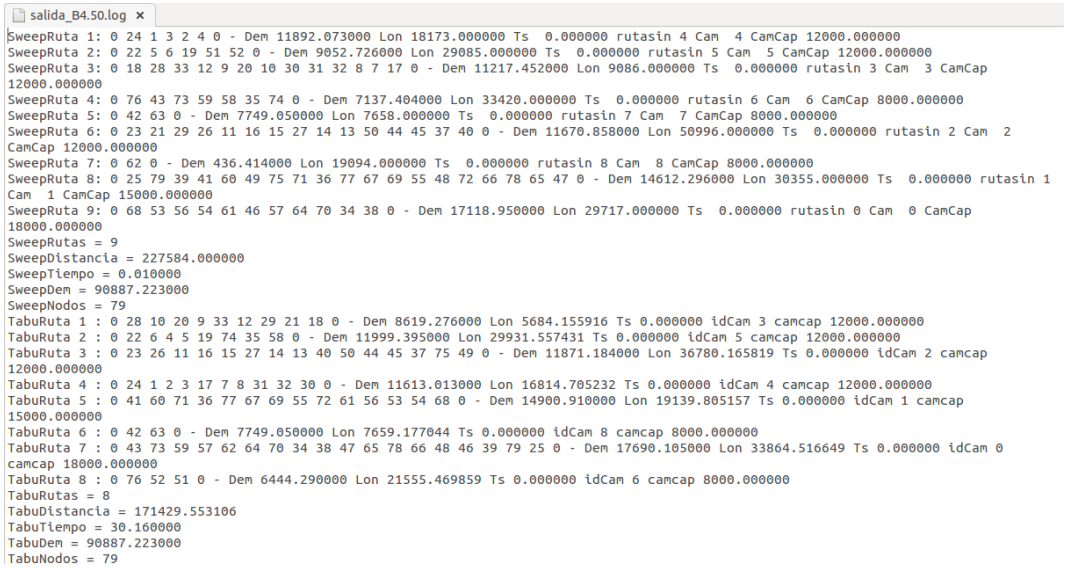
El algoritmo fue diseñado en C++, uno de los leguajes de alto nivel más utilizados por los programadores (Joyanes A, 2003). Comúnmente, los programas en C++ pasan a través de seis fases: edición, preprocesamiento, compilación, enlace, carga y ejecución (Deitel & Deitel, 2008). Las cuales se realizaron en el editor de textos Gedit y el compilador GNU de C de Linux. La elección de este lenguaje de programación se basó en la experiencia con la que se contaba en este lenguaje y la facilidad del mismo para desarrollar sistemas orientados a objetos. El desarrollo de la tesis pasó por las etapas típicas de cualquier proyecto. En primer lugar se realizó el levantamiento de las necesidades reales del programa y en la segunda etapa se efectuó el análisis y el diseño comenzando por diagramas más generales y concluyendo hacia los más particulares.

5.4.5 Archivo de Salida

Cuando termina de ejecutarse el algoritmo se crea un archivo llamado salida.log, que contiene los resultados del caso estudio. En la **figura 5.4** se muestra la una captura de pantalla del documento. El formato del archivo es el siguiente:

- ✓ **SweepRuta n - TabuRuta n**: Es la secuencia de clientes para la ruta n, además se muestra la demanda total atendida (Dem), la distancia recorrida en la ruta (Lon) y el vehículo que atendió la ruta (CamCap).
- ✓ **SweepRutas - TabuRutas**: Muestra el total de rutas creadas con el método de barrido y GTS respectivamente, en Barrido la cantidad de rutas puede exceder el número de camiones que se fijaron para el caso estudio, tal y como se explicó en las secciones anteriores.
- ✓ **SweepDistancia - TabuDistancia**: Es la distancia total recorrida por todas las rutas en cada método.
- ✓ **SweepTiempo - TabuTiempo**: Es el tiempo computacional empleado para dar la respuesta.
- ✓ **SweepDem - TabuDem:** Es la demanda total atendida.
- ✓ **SweepNodos - TabuNodos**: Son los nodos recorridos, sin incluir el depósito.

Figura 5.4 Captura de pantalla del archivo salida.log.


salida_B4.50.log ×

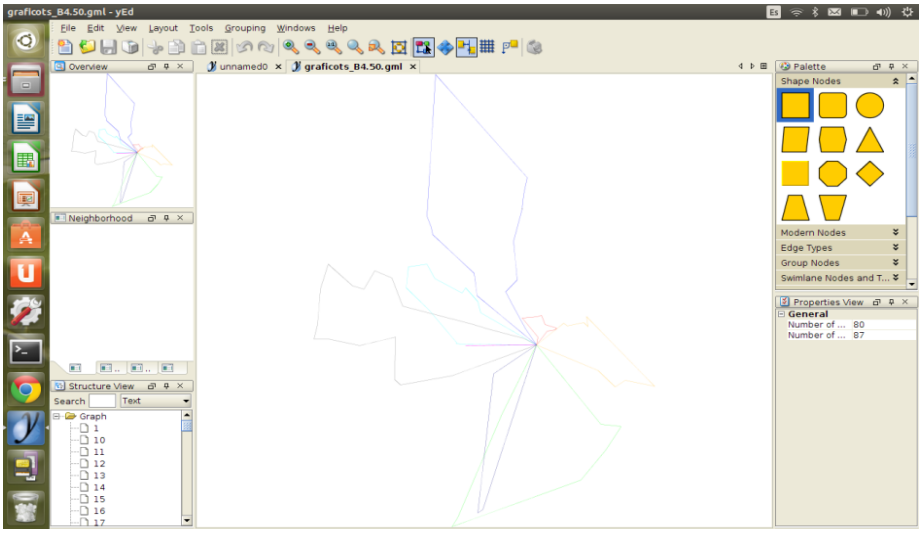
```
SweepRuta 1: 0 24 1 3 2 4 0 - Dem 11892.073000 Lon 18173.000000 Ts  0.000000 rutasin 4 Cam  4 CamCap 12000.000000
SweepRuta 2: 0 22 5 6 19 51 52 0 - Dem 9052.726000 Lon 29085.000000 Ts  0.000000 rutasin 5 Cam  5 CamCap 12000.000000
SweepRuta 3: 0 18 28 33 12 9 20 10 30 31 32 8 7 17 0 - Dem 11217.452000 Lon 9086.000000 Ts  0.000000 rutasin 3 Cam  3 CamCap
12000.000000
SweepRuta 4: 0 76 43 73 59 58 35 74 0 - Dem 7137.404000 Lon 33420.000000 Ts  0.000000 rutasin 6 Cam  6 CamCap 8000.000000
SweepRuta 5: 0 42 63 0 - Dem 7749.050000 Lon 7658.000000 Ts  0.000000 rutasin 7 Cam  7 CamCap 8000.000000
SweepRuta 6: 0 23 21 29 26 11 16 15 27 14 13 50 44 45 37 40 0 - Dem 11670.858000 Lon 50996.000000 Ts  0.000000 rutasin 2 Cam  2
CamCap 12000.000000
SweepRuta 7: 0 62 0 - Dem 436.414000 Lon 19094.000000 Ts  0.000000 rutasin 8 Cam  8 CamCap 8000.000000
SweepRuta 8: 0 25 79 39 41 60 49 75 71 36 77 67 69 55 48 72 66 78 65 47 0 - Dem 14612.296000 Lon 30355.000000 Ts  0.000000 rutasin 1
Cam  1 CamCap 15000.000000
SweepRuta 9: 0 68 53 56 54 61 46 57 64 70 34 38 0 - Dem 17118.950000 Lon 29717.000000 Ts  0.000000 rutasin 0 Cam  0 CamCap
18000.000000
SweepRutas = 9
SweepDistancia = 227584.000000
SweepTiempo = 0.010000
SweepDem = 90887.223000
SweepNodos = 79
TabuRuta 1 : 0 28 10 20 9 33 12 29 21 18 0 - Dem 8619.276000 Lon 5684.155916 Ts 0.000000 idCam 3 camcap 12000.000000
TabuRuta 2 : 0 22 6 4 5 19 74 35 58 0 - Dem 11999.395000 Lon 29931.557431 Ts 0.000000 idCam 5 camcap 12000.000000
TabuRuta 3 : 0 23 26 11 16 15 27 14 13 40 50 44 45 37 75 49 0 - Dem 11871.184000 Lon 36780.165819 Ts 0.000000 idCam 2 camcap
12000.000000
TabuRuta 4 : 0 24 1 2 3 17 7 8 31 32 30 0 - Dem 11613.013000 Lon 16814.705232 Ts 0.000000 idCam 4 camcap 12000.000000
TabuRuta 5 : 0 41 60 71 36 77 67 69 55 72 61 56 53 54 68 0 - Dem 14900.910000 Lon 19139.805157 Ts 0.000000 idCam 1 camcap
15000.000000
TabuRuta 6 : 0 42 63 0 - Dem 7749.050000 Lon 7659.177044 Ts 0.000000 idCam 8 camcap 8000.000000
TabuRuta 7 : 0 43 73 59 57 62 64 70 34 38 47 65 78 66 48 46 39 79 25 0 - Dem 17690.105000 Lon 33864.516649 Ts 0.000000 idCam 0
camcap 18000.000000
TabuRuta 8 : 0 76 52 51 0 - Dem 6444.290000 Lon 21555.469859 Ts 0.000000 idCam 6 camcap 8000.000000
TabuRutas = 8
TabuDistancia = 171429.553106
TabuTiempo = 30.160000
TabuDem = 90887.223000
TabuNodos = 79
```

Fuente: Salida del compilador GNU de C de Linux.

5.4.6 Gráficas de Salida

Para mostrar los resultados de manera gráfica se empleó una secuencia algorítmica reciclada de (Cplusplus.com, 2013), este código grafica las coordenadas X y Y de cada cliente y une los puntos de acuerdo a la secuencia generada por cada algoritmo, para el algoritmo de Barrido crea un archivo llamado **graficosw.gml** y para el GTS crea **graficots.gml**. El programa para visualizar los archivos se llama **yEd Graph Editor** versión **3.11.1,** yEd es una aplicación de escritorio poderosa que se puede utilizar para generar con rapidez y eficacia diagramas de alta calidad. Crea diagramas manualmente o importar los datos externos para el análisis. yEd está disponible gratuitamente y se ejecuta en todas las plataformas: Windows, Unix / Linux y Mac OS X**,** además soporta una amplia variedad de tipos de diagramas, diagramas de flujo, árboles genealógicos, redes semánticas, diagrama de clases UML, Redes Sociales entre otros (Yworks, 2013). En la **figura 5.5** se muestra una captura de pantalla del archivo graficots.gml.

Figura 5.5 Captura de pantalla del archivo graficots.gml.



Fuente: Salida de yEd Graphs Editor.

5.4.7 Hardware y Sistema Operativo

Durante el desarrollo y las pruebas se utilizó una PC Hp Pavilion dv4-2025la con procesador Intel Core i3 a 2.13 GHz con 4 GB de memoria RAM y disco duro de 500 GB. El sistema operativo sobre el cual se trabajó fue Ubuntu 13.10.

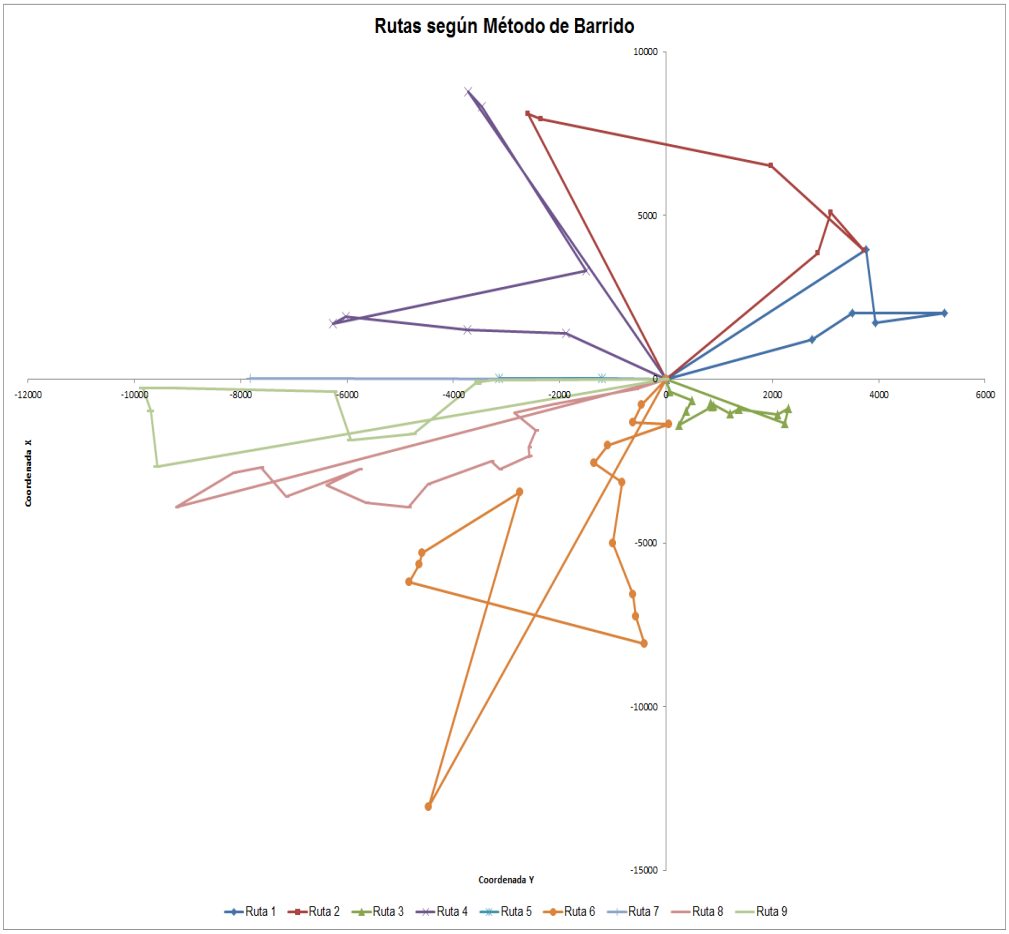
6. RESULTADOS COMPUTACIONALES.

En esta sección se presenta los resultados obtenidos con el algoritmo para el caso estudio, además de las pruebas realizadas según los diferentes parámetros que se consideraron.

6.1 Resultados de la Solución Inicial

En la **figura 6.1** se muestran las rutas creadas por la solución inicial, estas rutas no se estudiarán a fondo ya que la importancia recaerá en las rutas de la Búsqueda Tabú Granular.

Figura 6.1 Rutas de la solución inicial.



Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

Como se puede apreciar en la figura 6.1, algunas de las rutas presentan cruces consigo mismas, un factor que podría considerarse no deseado o ineficiente en el diseño de rutas, tal y como se muestra en (Vidal, 2010), pero ésta es una de las consecuencias cuando se prefiere simplicidad y rapidez en las respuestas, en lugar de complejidad y calidad en las mismas. En este contexto, es necesario recalcar que la secuenciación del método de barrido se realizó por medio del algoritmo del vecino más cercano, una técnica muy sencilla de emplear, pero que se considera miope, ya que, en una iteración escoge la mejor opción disponible sin "ver" que esto puede obligar a realizar malas elecciones en iteraciones posteriores, tal y como se puede apreciar en la **ruta 6** de la figura 6.1, donde los clientes se unen debido a su menor distancia inmediata, pero al final quedan clientes con largos trayectos que deben ser insertados generando así cruces en la ruta.

El algoritmo escogido para la solución inicial, tiene su justificación cuando se refiere al trabajo de Toth y Vigo (2003). En su documento aclaran el hecho de que la solución inicial puede o no ser factible, en cuanto a la cantidad de camiones se refiere, ya que al emplear la búsqueda tabú granular se eliminan las rutas menos cargadas, y los clientes son ubicados en mejores posiciones, así al final se obtienen muy buenos resultados sin que sea muy relevante el método empleado para generar la solución inicial.

En la **tabla 6.1** se muestran los valores que forman parte de la solución inicial. Como se puede apreciar en la tabla, el número de rutas creadas por el método de Barrido (**9**) excede la cantidad de camiones disponibles en el caso estudio, la cual es **8**. La distancia total recorrida fue de **227.584 m,** el promedio de utilización en cuanto a capacidad de los vehículos fue del **83.26%** y el tiempo computacional para obtener la respuesta fue de **0,01** segundos. El porcentaje de utilización de capacidad hace referencia a la relación entre la capacidad del camión y la demanda atendida en la ruta. Es importante notar que para algunas rutas este indicador es muy bajo, ya que la distribución de clientes en las rutas no es el mejor.
En la **tabla 6.2** se muestran las características de las rutas, donde para el cálculo del tiempo de recorrido se dividió la distancia recorrida entre la velocidad promedio; el número de paradas es equivalente a los clientes en cada ruta; el tiempo promedio de parada es el tiempo de parada en cada cliente multiplicado por el número de clientes; el tiempo promedio administrativo es tiempo administrativo en cada cliente multiplicado por el número de clientes, y el tiempo total de ruta es la suma de los 3 tiempos anteriores. En la tabla 6.2 para las rutas

3, 6, 8 y 9 el tiempo total empleado excede las 8 horas laborales permitidas por lo tanto a los conductores se les deben pagar horas extras. Además se emplea un camión adicional, lo cual incrementa el costo de esta solución.

Tabla 6.1 Datos de la solución inicial.

Resultados del Método Barrido					
Ruta	Camion (Kg)	Secuencia de Ruta	Costo de Ruta (m)	Demanda de Ruta (Kg)	Utilización Capacidad (%)
1	12.000	0 24 1 3 2 4 0	18.173,00	11.892,07	99,10
2	12.000	0 22 5 6 19 51 52 0	29.085,00	9.052,73	75,44
3	12.000	0 18 28 33 12 9 20 10 30 31 32 8 7 17 0	9.086,00	11.217,45	93,48
4	8.000	0 76 43 73 59 58 35 74 0	33.420,00	7.137,40	89,22
5	8.000	0 42 63 0	7.658,00	7.749,05	96,86
6	12.000	0 23 21 29 26 11 16 15 27 14 13 50 44 45 37 40 0	50.996,00	11.670,86	97,26
7	8.000	0 62 0	19.094,00	436,41	5,46
8	15.000	0 25 79 39 41 60 49 75 71 36 77 67 69 55 48 72 66 78 65 47 0	30.355,00	14.612,30	97,42
9	18.000	0 68 53 56 54 61 46 57 64 70 34 38 0	29.717,00	17.118,95	95,11
CapCam	105.000	Costo Total del Método	227.584,00	90.887,22	83,26

Fuente: Elaboración propia.

Tabla 6.2 Características de las rutas creadas por el método de barrido.

Ruta	Distancia Recorrida (Km)	Velocidad Promedio (Km/h)	Tiempo Recorrido (h)	Número de Paradas	Tiempo Promedio Parada (h)	Tiempo Administrativo (h)	Tiempo Total de Ruta (h)
1	18,17	45	0,40	5	2,5	1,25	4,15
2	29,09	45	0,65	6	3,0	1,50	5,15
3	9,09	45	0,20	13	6,5	3,25	9,95
4	33,42	45	0,74	7	3,5	1,75	5,99
5	7,66	45	0,17	2	1,0	0,50	1,67
6	51,00	45	1,13	15	7,5	3,75	12,38
7	19,09	45	0,42	1	0,5	0,25	1,17
8	30,36	45	0,67	19	9,5	4,75	14,92
9	29,72	45	0,66	11	5,5	2,75	8,91
Total	**227,58**		**5,06**	**79**	**39,50**	**19,75**	**64,31**

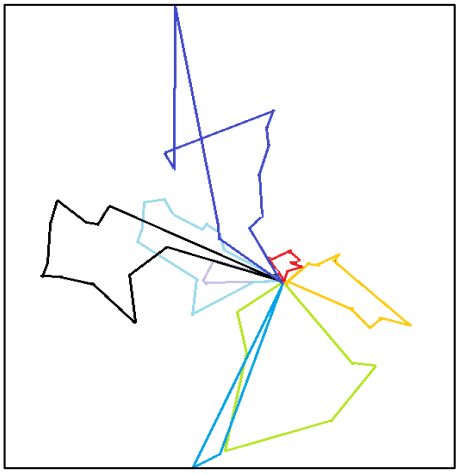
Fuente: Elaboración propia.

6.2 Análisis de Sensibilidad para β y $Maxiter$

La calibración de parámetros y experimentos computacionales son pasos importantes en el desarrollo de cualquier algoritmo. Esto es particularmente cierto en el caso de TS, ya que el número de parámetros requeridos por la mayoría de las implementaciones es bastante grande (Laguna, 1994) y (Gendreau, 2002). En este sentido, los parámetros a emplear en TS deben ser objeto de pruebas computacionales, con el fin de adaptarse al problema propuesto. Por tal motivo antes de establecer los valores para el parámetro de dispersión β y el número máximo de iteraciones $Maxiter$, se realizó una corrida de prueba con valores sugeridos en la bibliografía, para β se tomó un valor de 1.5 y el $Maxiter$ de $20u$, para este último parámetro 20u significa que se realizará 20 iteraciones por cada

cliente, este valor se tomó con el fin de que el algoritmo realice suficientes iteraciones hasta encontrar una buena solución. Además se definió un valor de 7 como periodo tabú ya que al variarlo entre 5 y 10 los impactos en la salida no fueron muy relevantes. La gráfica de salida fue la que se muestra en la **figura 6.2.**

Figura 6.2 Corrida de prueba para la solución final.



Fuente: Salida de yEd Graphs Editor.

Los resultados luego de realizar la corrida fueron: 8 rutas creadas con una distancia total recorrida de **187.573 m** y un tiempo computacional de 7,05 segundos. Comparando estos datos con la solución inicial, podemos encontrar una mejora de **40.011** m, es decir, una mejora del **17,58%.** Los resultados anteriores permitieron tener un punto de partida para calibrar el parámetro de dispersión y el número máximo de iteraciones.

Cuando se modifica β se puede incluir de manera sencilla aspectos de intensificación y diversificación en la búsqueda. Por ejemplo, mediante la modificación del parámetro de dispersión β, el número de arcos incluidos actualmente en el gráfico disperso se altera, ya que la regla de filtrado para los arcos se ha modificado, por lo tanto, una nueva solución posiblemente diferente se obtiene al final de la búsqueda. De acuerdo con Toth y Vigo (2003), el valor de β puede ser modificado dentro de un rango de 0,5 y 5,0 para lo cual se obtuvo los resultados mostrados en la **Tabla 6.3** y **Figura 6.3 a, b**.

En la tabla 6.3 se muestran los resultados obtenidos para la variación de β, como se puede apreciar la salida del método de barrido no se ve afectada, ya que su

función objetivo no depende del parámetro de dispersión. Por otro lado la salida del GTS cambio a medida que iba variando β, incluso el tiempo computacional se vio afectado proporcionalmente.

Tabla 6.3 Variación de β con Maxiter = 20u

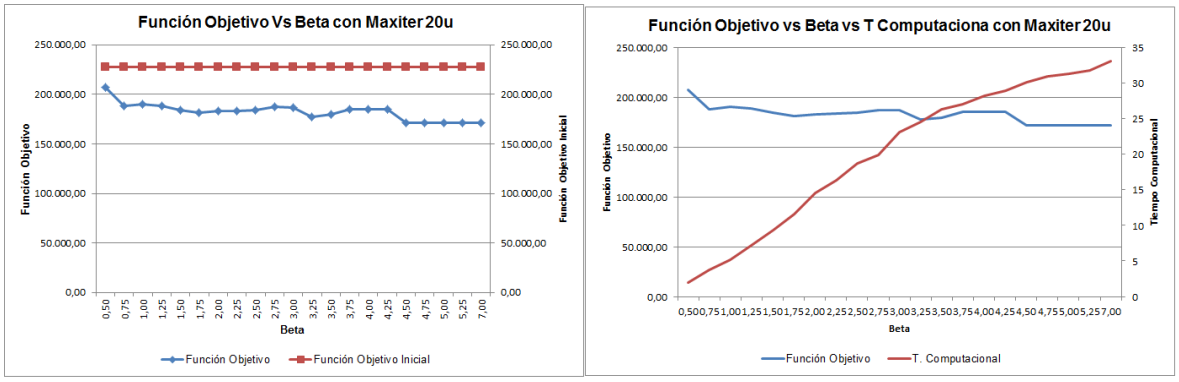
Resultados Sweep - GTS *Variación de Beta con Maxiter 20u*						
	Sweep			GTS		
Beta	Función Objetivo Inicial	N° de Rutas	Tiempo Computacional	Función Objetivo Final	N° de Rutas	Tiempo Computacional
0,50	227.584,97	9	0,01	207.531,97	8	1,95
0,75	227.584,97	9	0,01	188.213,78	8	3,73
1,00	227.584,97	9	0,01	190.149,21	8	5,12
1,25	227.584,97	9	0,01	188.758,85	8	7,23
1,50	227.584,97	9	0,01	184.332,21	8	9,34
1,75	227.584,97	9	0,01	181.149,22	8	11,58
2,00	227.584,97	9	0,01	182.901,42	8	14,53
2,25	227.584,97	9	0,01	183.302,31	8	16,39
2,50	227.584,97	9	0,01	184.423,40	8	18,65
2,75	227.584,97	9	0,01	187.131,65	8	19,89
3,00	227.584,97	9	0,01	186.998,67	8	23,14
3,25	227.584,97	9	0,01	177.411,97	8	24,53
3,50	227.584,97	9	0,01	179.724,51	8	26,25
3,75	227.584,97	9	0,01	185.298,75	8	27,06
4,00	227.584,97	9	0,01	185.298,75	8	28,21
4,25	227.584,97	9	0,01	185.298,75	8	28,96
4,50	227.584,97	9	0,01	171.429,55	8	30,16
4,75	227.584,97	9	0,01	171.429,55	8	30,91
5,00	227.584,97	9	0,01	171.429,55	8	31,32
5,25	227.584,97	9	0,01	171.429,55	8	31,81
7,00	227.584,97	9	0,01	171.429,55	8	33,10

Fuente: Elaboración propia.

La **figura 6.3a** muestra como la función objetivo del GTS (curva azul) se va alejando de la función objetivo del Barrido (curva roja), a medida que se va incrementando el valor de β, es decir, para este caso estudio la solución final se va mejorando con el incremento del parámetro de dispersión, aunque esta relación no es estrictamente directa ya que como se puede apreciar existen incrementos y decrementos a medida que sube β, como por ejemplo en el rango $[3.0,3.5]$ de la figura 6.3.

Por otro lado la **figura 6.3b** muestra cómo se relacionan la solución final, el parámetro de dispersión y el tiempo computacional requerido para terminar la ejecución del algoritmo. De la tabla 6.3 y la figura 6.3b queda claro que a medida que se incrementa β, se incrementa el tiempo requerido para obtener una buena solución, por lo tanto se puede concluir que al emplear un parámetro de dispersión grande, se pueden obtener mejores soluciones pero a dispensas de gastar tiempos computacionales mayores. Esto sucede porque al usar un β grande la búsqueda tabú se diversifica, es decir, tiene en cuenta arcos más largos o arcos que no había considerado antes.

Figura 6.3 a) Función Objetivo Vs Beta. **b)** Función Objetivo Vs Beta Vs T. Computacional



Fuente: Elaboración propia.

De la figura 6.3b también se podría escoger una buena solución con un tiempo computacional relativamente corto, este valor de solución se obtiene en el punto de intersección de la dos curvas $(\beta = 3.25, f(x) = 177.411{,}97\ y\ T.Comp = 24.53\ seg)$, aunque en este caso estudio se optó por escoger la mejor solución final independiente del tiempo computacional requerido, este valor se obtiene con un $\boldsymbol{\beta = 4.50}$, donde la función objetivo equivale a **171.429,55 m.**

Una vez determinado el parámetro de dispersión se pasó a seleccionar el mejor $Maxiter$, esto con el fin de encontrar una mejor solución final (de ser posible) al variar el máximo número de iteraciones. A continuación se muestra la tabla y las gráficas con el rango de valores analizados. En la **tabla 6.4** se presentan los resultados al variar el valor $Maxiter$ desde un valor de 2 a 30, dando como resultado una alteración en el valor de la solución final, lógicamente el tiempo computacional se incrementa a medida que se incremente el número de iteraciones por clientes.

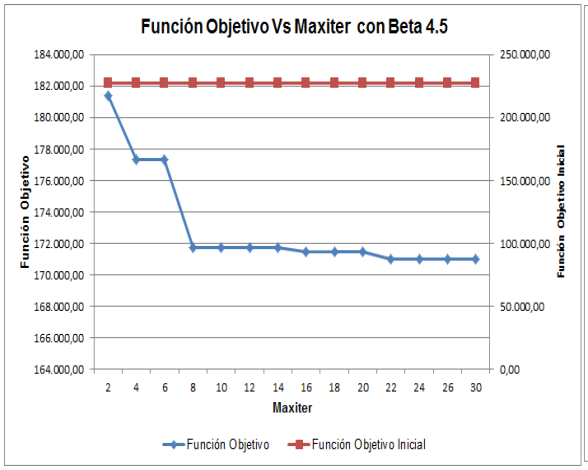
En la **figura 6.4a** se ve el comportamiento de la función objetivo a medida que se varia el número de iteraciones, como era de esperarse a medida que se incrementa el $maxiter$, el valor de la solución final mejora (curva azul) con respecto a la solución inicial (curva roja), el rango que presenta los mejores valores para la salidad del GTS, está entre $(8u, 22u)$, por lo tanto el valor para el $maxiter$ debe estar en ese rango.

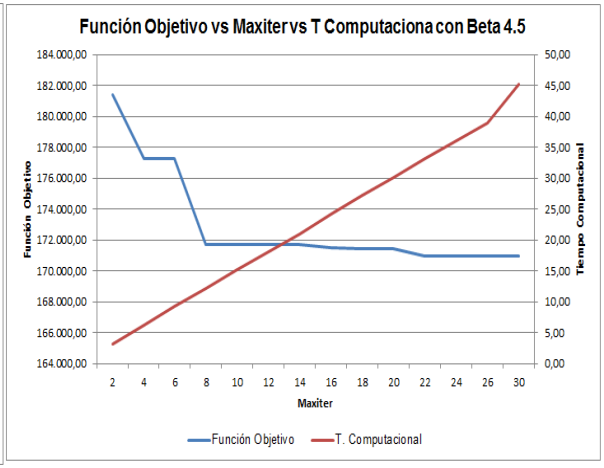
Tabla 6.4 Variación de Maxiter con β = 4.5

Resultados Sweep - GTS *Variación de Maxiter con Beta 4.50*						
	Sweep			GTS		
Maxiter	Función Objetivo Inicial	N° de Rutas	Tiempo Computacional	Función Objetivo Final	N° de Rutas	Tiempo Computacional
2	227.584,97	9	0,01	181.393,72	8	3,07
4	227.584,97	9	0,01	177.293,91	8	6,26
6	227.584,97	9	0,01	177.293,91	8	9,22
8	227.584,97	9	0,01	171.715,27	8	12,20
10	227.584,97	9	0,01	171.715,27	8	15,17
12	227.584,97	9	0,01	171.715,27	8	18,09
14	227.584,97	9	0,01	171.715,27	8	21,00
16	227.584,97	9	0,01	171.487,30	8	24,16
18	227.584,97	9	0,01	171.429,55	8	27,27
20	227.584,97	9	0,01	171.429,55	8	30,18
22	227.584,97	9	0,01	170.987,17	8	33,10
24	227.584,97	9	0,01	170.987,17	8	36,08
26	227.584,97	9	0,01	170.987,17	8	38,94
30	227.584,97	9	0,01	170.987,17	8	45,13

Fuente: Elaboración propia.

Figura 6.4 a) Función Objetivo Vs Maxiter. **b)** Función Objetivo Vs Maxiter Vs T. Computacional





Fuente: Elaboración propia

En la **figura 6.4b** se aprecia como el valor de la solución del GTS mejora a medida que se incrementa el número de iteraciones, pero lógicamente sacrificando tiempo de cómputo. El punto de cruce de las dos curvas $(Maxiter = 13u, f(x) = 171.715{,}27\ y\ T.Comp = 19.55\ seg)$ muestra una solución final con un buen valor para la función objetivo y un relativamente pequeño valor para el $maxiter$, lo que representaria un tiempo computacional más corto. Pero como se dijo anteriormente, el objetivo del caso estudio es obtener una solución del GTS con el mejor valor posible, lo cual representaría menores costos asociados al transporte;

este valor para el máximo número de iteraciones es $\boldsymbol{Maxiter = 22u}$, lo cual da como función objetivo un valor de **170.987,18 m.**

Finalmente los valores seleccionados para dar respuesta al caso estudio fueron $\beta = 4.50$, $Maxiter = 22u$ y periodo tabú de 7.

6.3 Resultados del GTS

En esta subsección se mostrarán la secuencia de rutas y las gráficas correspondientes a dichas rutas.

En la **tabla 6.5** se muestran los valores que forman parte de la solución final. El GTS generó 8 rutas, empleando únicamente los camiones que se tenían disponibles para el caso estudio. La distancia total recorrida fue de **170.987,18 m**, el promedio de utilización en cuanto a capacidad de los vehículos fue del **92.82%** y el tiempo computacional para obtener la respuesta fue de 33,10 segundos.

En la **tabla 6.6** se comparan las respuestas de la solución inicial con la solución final, se puede ver que hay una mejora en la distancia total recorrida de **56.596,82m**, equivalente a una disminución del **24.87%** en el costo de rutas, a pesar de que el tiempo de cómputo es mayor para la salida GTS. También es relevante señalar que la salida GTS presenta un mejor porcentaje de utilización en cuanto a capacidad de los vehículos, mejorando la distribución de clientes por ruta en un **9,56%.**

Tabla 6.5 Datos de la solución final.

Resultados del Método Búsqueda Tabú Granular					
Ruta	**Camion (Kg)**	**Secuencia de Ruta**	**Costo de Ruta (m)**	**Demanda de Ruta (Kg)**	**Utilización Capacidad (%)**
1	12.000	0 28 10 20 9 33 12 29 21 18 0	5.684,15	8.619,27	71,83
2	12.000	0 22 6 4 5 19 74 35 58 0	29.931,55	11.999,40	100,00
3	12.000	0 23 26 11 16 15 27 14 13 40 50 44 45 37 75 49 0	36.780,17	11.871,18	98,93
4	12.000	0 24 1 2 3 7 17 8 32 31 30 0	16.372,33	11.613,01	96,78
5	15.000	0 41 60 71 36 77 67 69 55 72 61 56 53 54 68 0	19.139,81	14.900,91	99,34
6	8.000	0 42 63 0	7.659,18	7.749,05	96,86
7	18.000	0 43 73 59 57 62 64 70 34 38 47 65 78 66 48 46 39 79 25 0	33.864,52	17.690,11	98,28
8	8.000	0 76 52 51 0	21.555,47	6.444,29	80,55
CapCam	**97.000**	**Costo Total del Método**	**170.987,18**	**90.887,22**	**92,82**

Fuente: Elaboración propia.

Tabla 6.6 Comparación de salidas por los dos métodos.

Resultados	Método de Barrido	Método GTS	Mejora
Numero de Rutas	9	8	1
Distancia Total (m)	227.584,00	170.987,18	56.596,82
Tiempo Computacional (S)	0,01	33,10	-33,09
Utilización Capacidad Promedio (%)	83,26	92,82	9,56

Fuente: Elaboración propia.

A continuación se presentan las gráficas para las rutas creadas por el GTS, cuyos datos se pueden ver en la tabla 6.5. Adicionalmente se presentarán las características de cada ruta (**tabla 6.7**), empleando los supuestos definidos en la sección 4.1.1.

En la tabla 6.7 para las rutas 3, 5 y 7 el tiempo total empleado excede las 8 horas laborales permitidas por lo tanto a los conductores se les deben pagar horas extras. Si se comparan las tablas 6.2 y 6.7 se puede ver que luego de haber ejecutado el algoritmo GTS se obtuvo una disminución en los costos de transporte, ya que no sólo se disminuyó en 1 la cantidad de vehículos empleados, sino que se mejoró la distribución de clientes por ruta, evitando el pago de horas extras en una ruta y finalmente lo más importante se disminuyó la distancia total recorrida, lo que significa un ahorro considerable en gastos como consumo de combustible y demás costos variables.

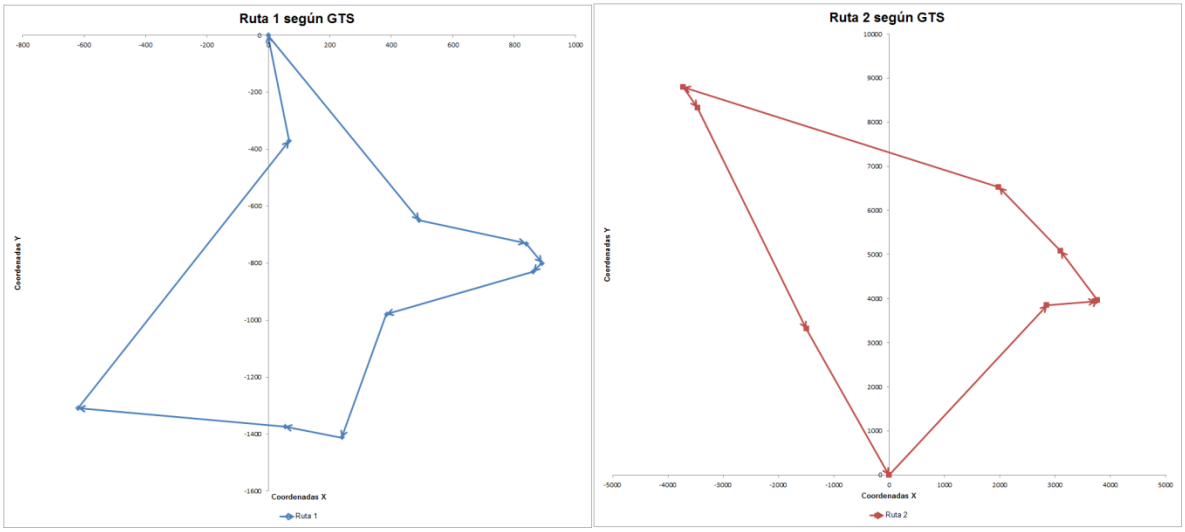
De las figuras 6.5 a 6.9 se muestran gráficamente las rutas, en las cuales no se encuentran cruces consigo mismas, indicando una eficiencia en su construcción.

Tabla 6.7 Características de las rutas.

Ruta	Distancia Recorrida (Km)	Velocidad Promedio (Km/h)	Tiempo Recorrido (h)	Número de Paradas	Tiempo Promedio Parada (h)	Tiempo Administrativo (h)	Tiempo Total de Ruta (h)
1	5,68	45	0,13	9	4,5	2,25	6,88
2	29,93	45	0,67	8	4,0	2,00	6,67
3	36,78	45	0,82	15	7,5	3,75	12,07
4	16,37	45	0,36	10	5,0	2,50	7,86
5	19,14	45	0,43	14	7,0	3,50	10,93
6	7,66	45	0,17	2	1,0	0,50	1,67
7	33,86	45	0,75	18	9,0	4,50	14,25
8	21,56	45	0,48	3	1,5	0,75	2,73
Total	**170,99**		**3,80**	**79**	**39,50**	**19,75**	**63,05**

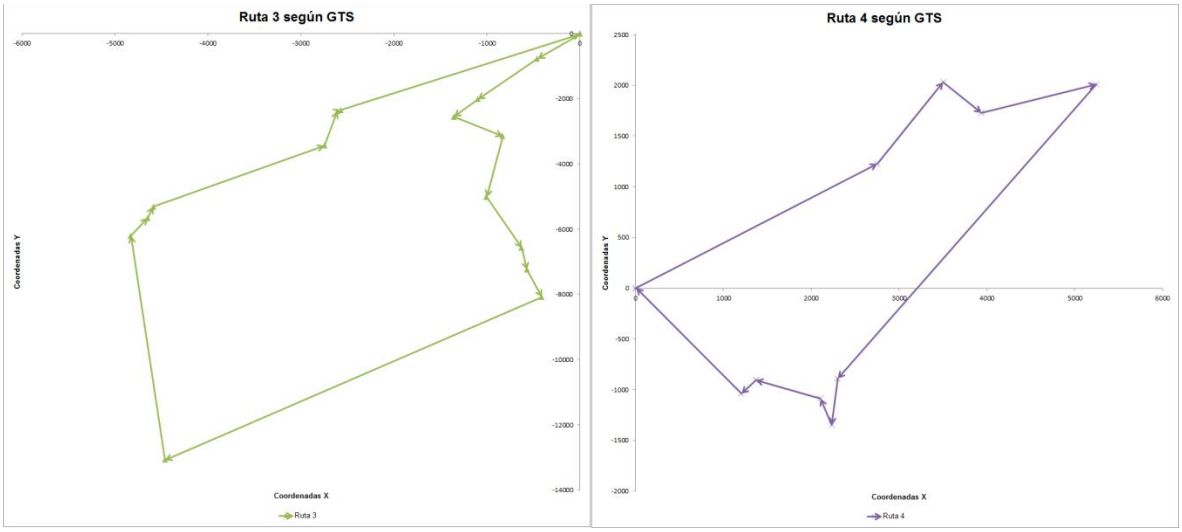
Fuente: Elaboración propia.

Figura 6.5 Gráficas de la Rutas 1 y 2.



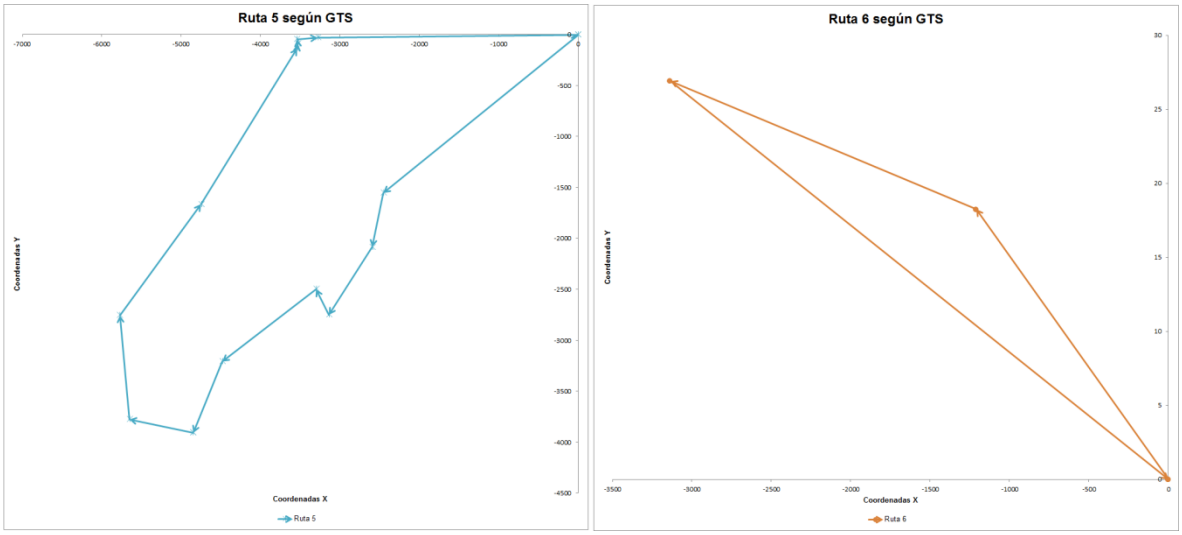
Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

Figura 6.6 Gráficas de las Rutas 3 y 4.



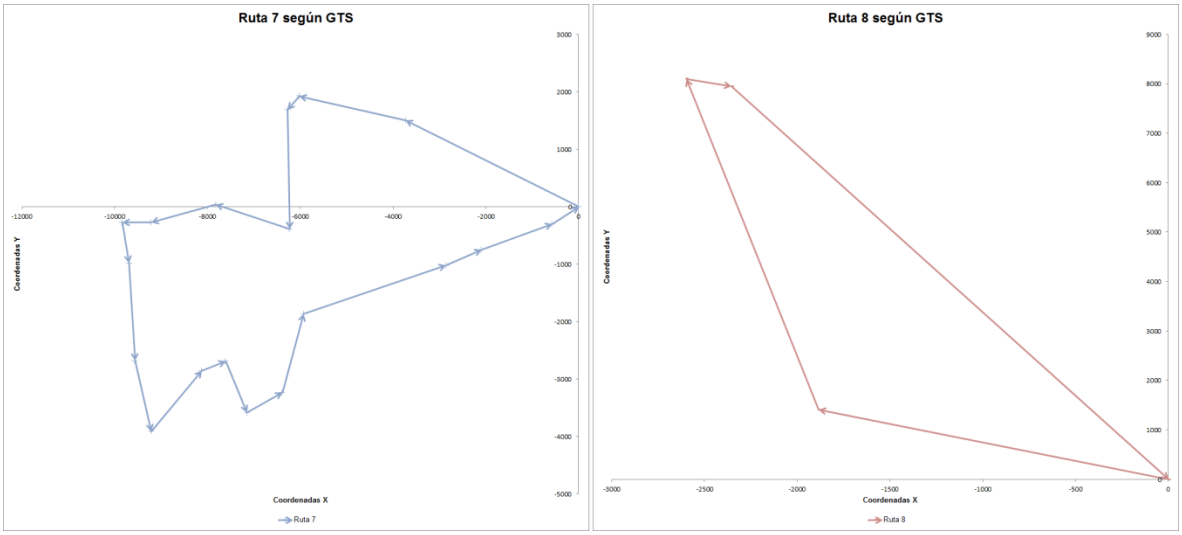
Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

Figura 6.7 Gráficas de las Rutas 5 y 6.



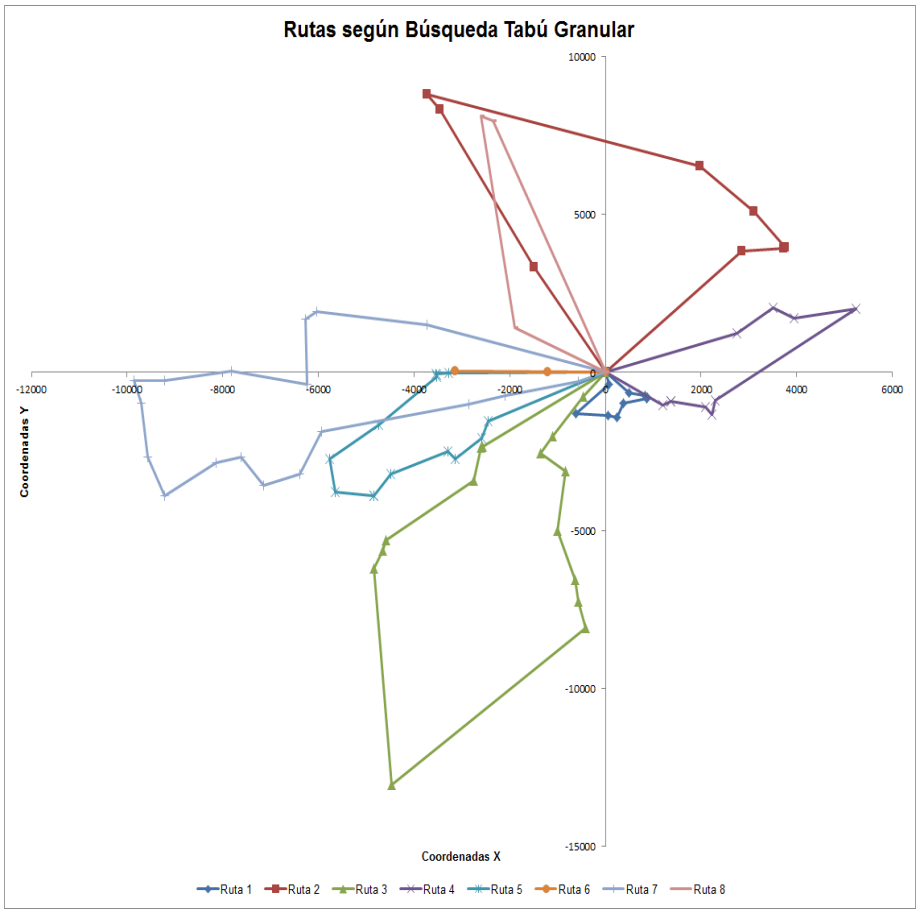
Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

Figura 6.8 Gráficas de la Rutas 7 y 8.



Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

Figura 6.9 Rutas de la solución final.



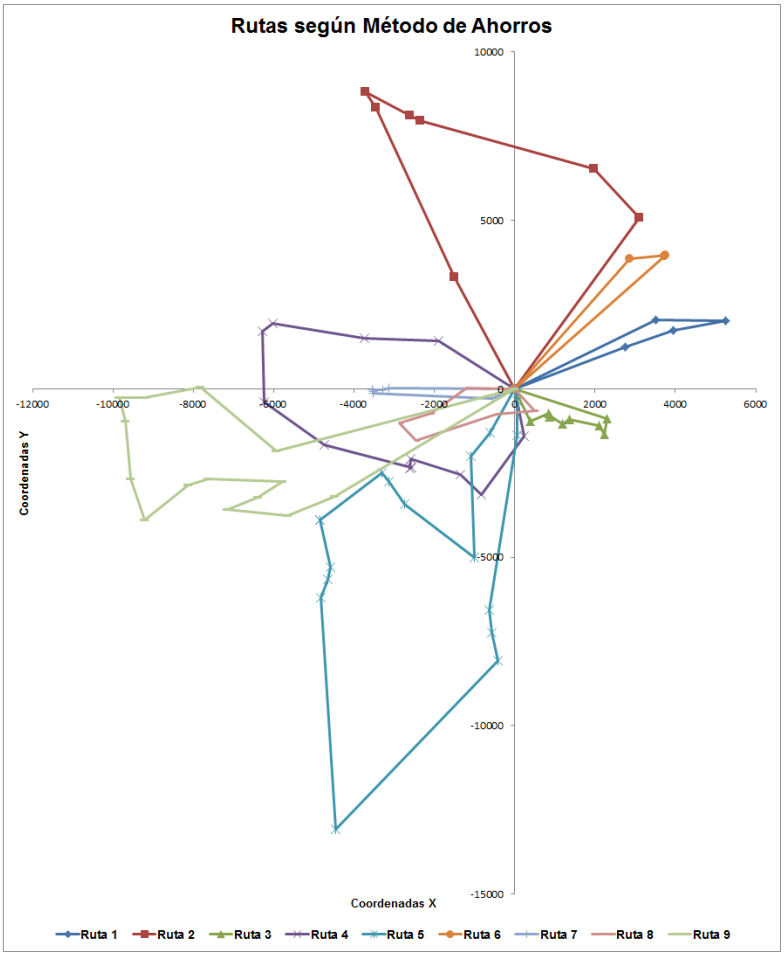
Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

6.4 Cambio de Método en la Solución Inicial

Una manera de comprobar la eficiencia del GTS y su comportamiento en el caso de estudio propuesto, es modificando la solución inicial, para ello se implementó el algoritmo de los ahorros de Clarke y Wright. Este método fue escogido ya que en general sus soluciones son mucho mejores a las del algoritmo de Barrido (Marti, 2004). El objetivo de esta subsección es probar la respuesta del GTS con una mejor solución inicial. A continuación se presentará brevemente dicha modificación.

En la **figura 6.10** se muestran las rutas creadas por la solución inicial, como se puede apreciar las 9 rutas creadas no presentan cruces consigo mismas, mejorando en este sentido a la salida del método de barrido.

Figura 6.10 Solución inicial generada por el Método de Ahorros.



Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

En la **tabla 6.8** se muestran los valores que forman parte de la solución inicial. Como se puede apreciar en la tabla, el número de rutas creadas por el método de ahorros es **9,** similar al algoritmo de Barrido. La distancia total recorrida fue de **182.606 m, 44.978 m** menos que la distancia del método de Barrido, lo cual comprueba que el algoritmo de los ahorros evaluado para este caso estudio en particular, ofrece una solución un poco mejor que el algoritmo de Barrido. El promedio de utilización en cuanto a capacidad de los vehículos fue del **85.73%,** solo un **2.47%** mejor que con el primer método, expresando una mejor distribución

de clientes por ruta y finalmente el tiempo computacional para obtener la respuesta fue de 0,08 segundos.

Al comparar las respuestas obtenidas en la solución inicial, mediante el método de Ahorros y Barrido, se hace necesario recordar que dichos algoritmos se diseñaron tal y como lo expresa la bibliografía, aunque a los dos se les adicionó una secuencia particular, que permite crear un vehículo adicional a los fijados con anterioridad en el inicio del caso estudio. Lo anterior se implementó con el fin de dar un cierto grado de flexibilidad en la búsqueda de una solución inicial, en otras palabras, se evitó que el algoritmo consumiera recursos computacionales, en una respuesta a la cual se le permite ser inviable desde el punto de vista de la capacidad de los camiones o la cantidad de los mismos, tal y como se expresa en (Toth & Vigo, 2003). Es por este motivo, que las respuestas de los métodos empleados en las soluciones iniciales difieran levemente a lo que se puede ver comúnmente en la bibliografía.

Tabla 6.8 Resultados del Método de Ahorros.

Resultados del Método Ahorros					
Ruta	Camion (Kg)	Secuencia de Ruta	Costo de Ruta (m)	Demanda de Ruta (Kg)	Utilización Capacidad (%)
1	8.000	0 1 3 2 24 0	13.936,69	7.232,07	90,40
2	12.000	0 5 19 51 52 74 35 58 0	28.679,65	11.128,67	92,74
3	12.000	0 7 17 8 31 32 30 20 9 10 33 0	7.672,05	7.353,94	61,28
4	12.000	0 12 16 11 60 75 49 61 57 59 73 43 76 0	23.321,13	11.182,91	93,19
5	15.000	0 21 26 15 37 71 36 69 67 45 44 50 40 13 14 27 29 18 0	42.268,87	14.607,44	97,38
6	8.000	0 22 4 6 0	13.624,68	6.616,73	82,71
7	12.000	0 25 56 53 54 68 63 0	8.844,04	10.919,39	90,99
8	8.000	0 28 23 41 39 79 42 0	9.485,66	5.972,39	74,65
9	18.000	0 77 55 66 48 72 78 65 47 38 34 70 64 62 46 0	34.773,23	15.873,68	88,19
CapCam	**105.000**	**Costo Total del Método**	**182.606,00**	**90.887,22**	**85,73**

Fuente: Elaboración propia.

Tabla 6.9 Características de las rutas generadas por método de los ahorros.

Ruta	Distancia Recorrida (Km)	Velocidad Promedio (Km/h)	Tiempo Recorrido (h)	Número de Paradas	Tiempo Promedio Parada (h)	Tiempo Administrativo (h)	Tiempo Total de Ruta (h)
1	13,94	45	0,31	4	2,0	1,00	3,31
2	28,68	45	0,64	7	3,5	1,75	5,89
3	7,67	45	0,17	10	5,0	2,50	7,67
4	23,32	45	0,52	12	6,0	3,00	9,52
5	42,27	45	0,94	17	8,5	4,25	13,69
6	13,62	45	0,30	3	1,5	0,75	2,55
7	8,84	45	0,20	6	3,0	1,50	4,70
8	9,49	45	0,21	6	3,0	1,50	4,71
9	34,77	45	0,77	14	7,0	3,50	11,27
Total	**182,61**		**4,06**	**79**	**39,50**	**19,75**	**63,31**

Fuente: Elaboración propia

En la **tabla 6.9** se aprecian las características de cada ruta creada por el algoritmo de los ahorros, es importante destacar que para esta solución se asume un costo adicional a la respuesta de este método ya que se emplean 9 camiones y no 8 como establece el problema originalmente. Además para las rutas 4, 5 y 9 se ve la necesidad de contratar horas extras ya se exceden la jornada normal de trabajo.

Al igual que se hizo en la subsección 6.2, el GTS tuvo que "calibrarse" o adaptarse al cambio de solución inicial, para ello se variaron los parámetros, dando como resultado $\beta = 2.0,\ Maxiter = 5u$ y periodo tabú de 10. Lo anterior se realizó debido a la naturaleza intrínseca de la metaheurística, ya que muchos procedimientos TS son extremadamente sensibles a los ajustes de las condiciones iniciales o parámetros, por lo tanto no es raro ver que el desempeño de un procedimiento mejorado, pueda sufrir modificaciones después de cambiar el valor de uno o dos parámetros clave (por desgracia, no siempre es obvio determinar qué parámetros son los más importantes en un procedimiento dado)(Gendreau, 2002). En este contexto, el ajuste de parámetros luego de cambiar la solución inicial es una falencia que posee toda metaheurística robusta, en este caso del TS, ya que esta fase forma parte de las características particulares del mismo. Por otro lado, al correr el programa con los parámetros iniciales se obtiene una solución final de muy buena calidad, tal y como puede apreciarse en la **tabla 6.10.**

Tabla 6.10 Análisis de sensibilidad para el refinamiento de los parámetros en el algoritmo CW-GTS

Resultados CW - GTS *Variación de Beta con Maxiter 20u*						
	CW			GTS		
Beta	Función Objetivo Inicial	N° de Rutas	Tiempo Computacional	Función Objetivo Final	N° de Rutas	Tiempo Computacional
0,50	182.606,01	9	0,06	182.606,01	9	0,13
0,75	182.606,01	9	0,08	177.823,26	8	0,15
1,00	182.606,01	9	0,06	177.519,53	8	0,18
1,25	182.606,01	9	0,06	176.054,99	8	0,26
1,50	182.606,01	9	0,06	176.054,99	8	0,67
1,75	182.606,01	9	0,06	176.054,99	8	0,78
2,00	182.606,01	9	0,06	**173.951,75**	8	**1,06**
2,25	182.606,01	9	0,06	174.212,50	8	1,35
2,50	182.606,01	9	0,06	174.212,50	8	1,61
2,75	182.606,01	9	0,06	174.212,50	8	2,03
3,00	182.606,01	9	0,06	173.951,75	8	2,13
3,25	182.606,01	9	0,06	173.951,75	8	2,17
3,50	182.606,01	9	0,06	173.951,75	8	2,42
3,75	182.606,01	9	0,06	174.212,50	8	2,71
4,00	182.606,01	9	0,06	174.212,50	8	2,96
4,25	182.606,01	9	0,06	174.212,50	8	3,06
4,50	182.606,01	9	0,06	**173.951,75**	8	**3,11**
4,75	182.606,01	9	0,06	173.951,75	8	3,17
5,00	182.606,01	9	0,06	173.951,75	8	3,25

Fuente: Elaboración propia

En la tabla 6.10 se puede ver que la salida del GTS empleando los parámetros usados con el método de Barrido, obtuvo la mejor respuesta encontrada usando el método de los ahorros como solución inicial $(\beta = 4.5\ y\ f(x) = 173.951{,}75)$, en otras palabras, se pudo haber ejecutado el algoritmo CW – GTS, sin la necesidad de cambiar el valor de los parámetros iniciales, pero es aquí donde se aplica lo expresado por Gendreau (2002), cuando se refiere a que en ciertas ocaciones los valores de los parámetros que son excelentes para las instancias a la mano, quizas no sean los óptimos cuando se modifican las condiciones iniciales de la instancia. Pero esto no significa que al emplear los mismos parámetros luego de cambiar las condiciones iniciales de una instancia, se obtengan soluciones de mala calidad, al contrario se siguen obteniendo muy buenas respuestas, sin embargo, si se quiere ser meticuloso se puede obtener unas respuestas un poco mejores, ya sea en cuanto a la calidad de la respuesta, o al tiempo computacional empleado para obtenerla. En lo que concierne a este caso estudio, los parámetros se refinaron una vez que se cambió de solución inicial, con el fin de obtener la misma respuesta que se habría encontrado con la primera configuración de los parámetros, pero en un tiempo computacional más corto. Si se quiere una descripción más detallada del proceso de calibración para un procedimiento TS, el lector debe referirse a (Crainic et al., 1993).

Una vez ajustado el GTS con los valores presentados anteriormente se obtuvo la respuesta presentada en la **tabla 6.11**. En esa tabla se puede apreciar que el GTS generó 8 rutas. La distancia total recorrida o la función objetivo final fue de **173.951,76 m**, el promedio de utilización en cuanto a capacidad de los vehículos fue del **91.67%** y el tiempo computacional para obtener la respuesta fue de 1,06 segundos. En la **tabla 6.12** se comparan las respuestas de la solución inicial con la solución final, se puede ver que hay una mejora en la distancia total recorrida de **8.654,24m**, equivalente a una disminución del **4,74%** en el costo de rutas. La mejora en cuanto a la distribución de clientes por ruta es de **5,94%.** En la **tabla 6.13** se presentan las características propias de cada ruta, este caso también se presentan horas adicionales a la jornada normal de trabajo, es el caso de las rutas 4, 5 y 8. Al igual que se hizo en la subsección 6.3, cuando se comparan las tablas 6.9 y 6.13 se puede concluir de manera global una disminución en los costos de transporte representado básicamente en la disminución de un vehículo empleado, mejora en la utilización de la capacidad y una reducción en la distancia total recorrida.

A continuación en las **figuras 6.11** y **6.12** se muestran gráficamente las rutas creadas por la Búsqueda Tabú Granular. Si se comparan las figuras 6.10 y 6.11 se

puede apreciar que los cambios realizados por el GTS son “pocos” ya que algunas rutas son prácticamente iguales, es el caso de las rutas 1, 2 y 3, el resto de rutas se modificó levemente debido a que el GTS eliminó la ruta 6 del método de ahorros y tuvo que reasignar esos clientes en otras rutas. También es importante notar que las nuevas rutas no presentan cruces consigo mismas, indicando una eficiencia en su construcción.

Tabla 6.11 Resultados del GTS con método de Ahorros como solución inicial.

Resultados del Método Búsqueda Tabú Granular					
Ruta	Camion (Kg)	Secuencia de Ruta	Costo de Ruta (m)	Demanda de Ruta (Kg)	Utilización Capacidad (%)
1	8.000	0 1 3 2 24 0	13.936,70	7.232,07	90,40
2	12.000	0 5 19 51 52 74 35 58 0	28.679,66	11.128,67	92,74
3	12.000	0 4 7 17 8 31 32 30 20 9 10 0	7.113,67	11.941,94	99,52
4	12.000	0 33 12 11 60 75 49 61 57 59 73 43 76 0	22.361,09	11.909,91	99,25
5	15.000	0 21 26 16 15 37 71 36 69 67 45 44 50 40 13 14 27 29 18 0	42.314,55	14.952,44	99,68
6	12.000	0 56 53 54 68 63 0	8.749,71	10.769,39	89,74
7	8.000	0 22 28 23 42 6 0	16.003,07	4.968,73	62,11
8	18.000	0 25 41 77 55 66 48 72 78 65 47 38 34 70 64 62 46 39 79 0	34.793,31	17.984,08	99,91
CapCam	**97.000**	**Costo Total del Método**	**173.951,76**	**90.887,23**	**91,67**

Fuente: Elaboración propia.

Tabla 6.12 Comparación de salidas por los Ahorros y GTS.

Resultados	Método de Ahorros	Método GTS	Mejora
Numero de Rutas	9	8	1
Distancia Total (m)	182.606,00	173.951,76	8.654,24
Tiempo Computacional (S)	0,08	1,06	-0,98
Utilización Capacidad Promedio	85,73	91,67	5,94

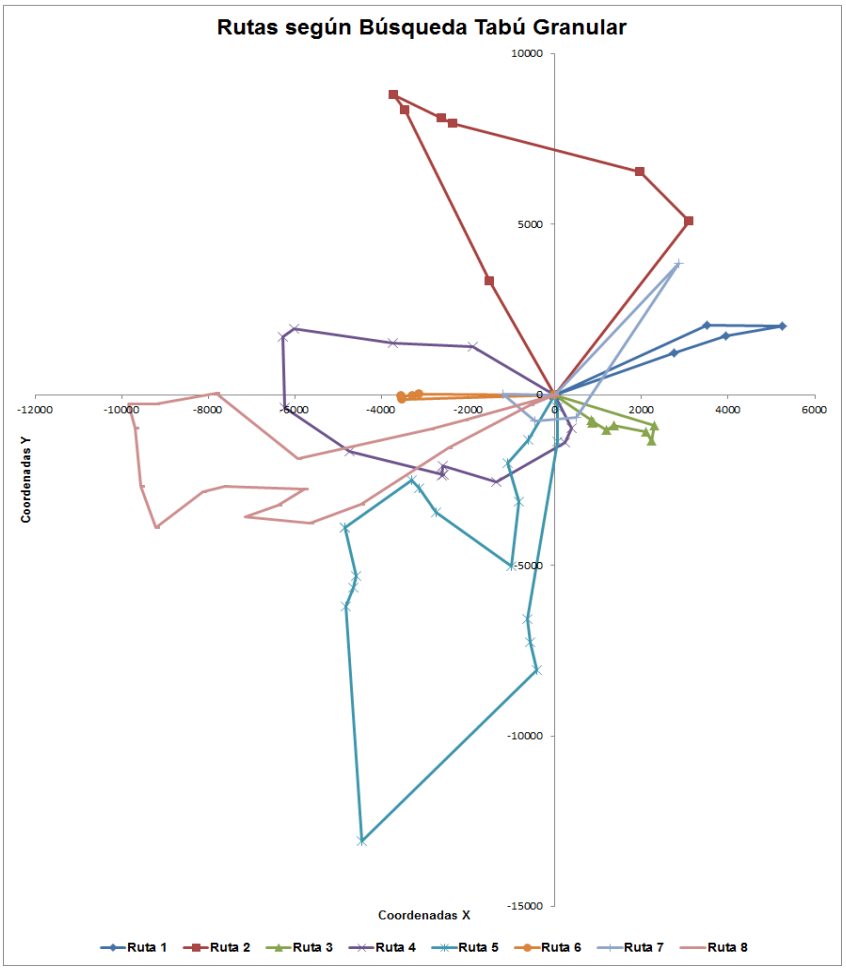
Fuente: Elaboración propia.

Tabla 6.13 Característica de las Rutas creadas.

Ruta	Distancia Recorrida (Km)	Velocidad Promedio (Km/h)	Tiempo Recorrido (h)	Número de Paradas	Tiempo Promedio Parada (h)	Tiempo Administrativo (h)	Tiempo Total de Ruta (h)
1	13,94	45	0,31	4	2,0	1,00	3,31
2	28,68	45	0,64	7	3,5	1,75	5,89
3	7,11	45	0,16	10	5,0	2,50	7,66
4	22,36	45	0,50	12	6,0	3,00	9,50
5	42,31	45	0,94	18	9,0	4,50	14,44
6	8,75	45	0,19	5	2,5	1,25	3,94
7	16,00	45	0,36	5	2,5	1,25	4,11
8	34,79	45	0,77	18	9,0	4,50	14,27
Total	**173,95**		**3,87**	**79**	**39,50**	**19,75**	**63,12**

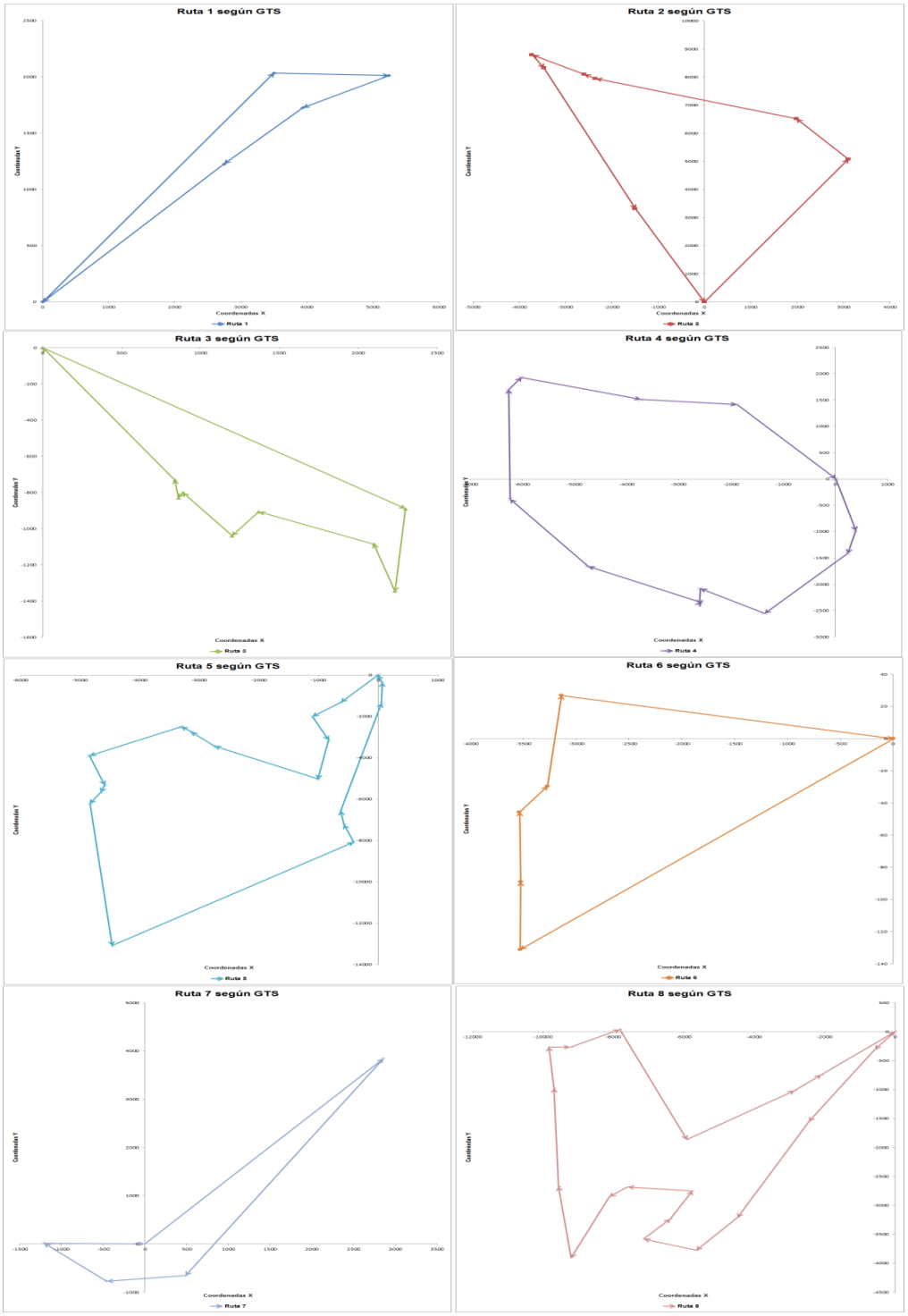
Fuente: Elaboración propia.

Figura 6.11 Rutas de la solución final creadas mediante el método de los ahorros como solución inicial.



Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

Figura 6.12 Individualización de las rutas de la solución final.



Fuente: Adoptado de la salida de yEd Graphs Editor y elaborado por el autor.

Una vez realizada esta variación se pudo comprobar la eficiencia del GTS y su respuesta frente a un mejor método empleado como solución inicial. Para puntualizar las diferencias en los resultados de las subsecciones 6.3 y 6.4 se presenta la siguiente tabla.

Tabla 6.14 Comparación del GTS frente a dos métodos en su solución inicial.

	Método de Barrido y GTS				Método de Ahorros y GTS			
	N° de Rutas	Función Objetivo (m)	Tiempo Computacional (S)	Utilización Capacidad Promedio (%)	N° de Rutas	Función Objetivo (m)	Tiempo Computacional (S)	Utilización Capacidad Promedio (%)
Solución Inicial	9	227.584,00	0,01	83,26	9	182.606,00	0,08	85,73
Solución Final	8	170.987,18	33,10	92,82	8	173.951,76	1,06	91,67
Mejora	1	56.596,82	-33,09	9,56	1	8.654,24	-0,98	5,94

Fuente: Elaboración propia.

En la **tabla 6.14** se puede apreciar que el algoritmo de ahorros generó una mejor solución inicial, pero el mejor valor de la solución final se obtuvo con el método de barrido. La mejora en la función objetivo presenta un valor más significativo con el primer algoritmo, puesto que su solución inicial no fue tan buena, por el contrario con el segundo método sólo se exhibe una mejora del 4,74%. El porcentaje de utilización promedio de la capacidad es 1,15% mejor en la solución final del primer método que la del segundo. Finalmente el mejor tiempo computacional para generar una solución final, fue el empleado por el segundo algoritmo.

Con base en lo anterior y de acuerdo a lo expresado anteriormente, este trabajo de grado optará por tomar la mejor solución final, independiente del tiempo computacional empleado, puesto que ese es el objetivo general planteado desde un principio, así pues se escogerá el algoritmo GTS implementado con el método de barrido como generador de la solución inicial. Considerando lo anterior y lo presentado en las subsecciones 6.3 y 6.4, específicamente la comparación de las tablas 6.2 con 6.7 y las tablas 6.9 con 6.13, se cubre el último objetivo específico de este trabajo de grado.

7. CONCLUSIONES Y TRABAJOS FUTUROS.

7.1 Conclusiones

En este trabajo de grado se implementó un algoritmo basado en Búsqueda Tabú, para solucionar un problema de ruteo de vehículos con flota heterogénea, esta variante del VRP es un poco más realista que la versión original puesto que la flota de vehículos de una compañía de camiones raramente es homogénea. Los vehículos difieren en sus equipos, capacidad de carga, edad y estructura de costos.

Para dar solución al problema de ruteo de vehículos heterogéneo existen diversos algoritmos en la literatura, como se pudo apreciar en el segundo capítulo de este trabajo. Sin embargo se decidió diseñar una metodología que utilizara el algoritmo de Búsqueda Tabú ya que es una de las mejores metaheurísticas para solucionar muchos problemas de optimización duros, a pesar de que los tiempos computacionales necesarios para detectar las mejores soluciones son generalmente muy grandes en comparación con los métodos tradicionales (Hillier & Lieberman, 2010). Es por esta razón por lo cual se decidió emplear una variante del TS, en esta versión de la Búsqueda Tabú se desarrolló una técnica efectiva de estrategias de listas de candidatos, el enfoque propuesto llamado Búsqueda Tabú Granular (GTS), se basa en el uso de vecindarios granulares, los cuales incluyen un pequeño número de movimientos "prometedores", que se evalúan en menos tiempo que los completos, por tal motivo se obtuvo buenas respuestas en tiempos de cómputo relativamente pequeños. Para este trabajo se empleó la estrategia de Toht y Vigo (2003), para obtener los vecindarios granulares.

Una de las ventajas de haber implementado vecindarios granulares fue la flexibilidad que le aportó al algoritmo, ya que al tratarse de un caso estudio con datos reales, la dispersión de los clientes se convirtió en una variable a vencer, es el caso de algunos clientes, los cuales se encontraban muy retirados del resto. En esta situación el incremento del parámetro de dispersión β permitió diversificar la búsqueda a movimientos los cuales no se habían considerado antes, con el fin de encontrar buenas soluciones, a expensas de requerir un mayor tiempo computacional, tal fue el caso de los resultados presentados con el algoritmo de barrido como solución inicial, donde se obtuvo un valor final de 170.987,18 m en 33,10 segundos para un $\beta = 4.5$. Lo anterior no establece que la calidad de la solución final es una función del incremento monótono del parámetro de

dispersión, es decir, del esfuerzo computacional en general, ya que como se pudo apreciar en la subsección 6.4, se obtuvo los mejores resultados con un valor de $\beta = 2.0$, dando una función objetivo final de 173.951,76 m en un tiempo computacional de 1,06 segundos. En definitiva el desempeño del algoritmo implementado no está ligado estrechamente al tipo de metodología empleada en la solución inicial, ni tampoco a valores fijos de los parámetros usados por el algoritmo, sino que al contar con una solución inicial aceptable y realizar experimentos computacionales con el fin de que el modelo creado se adapte a los datos, condiciones y restricciones del caso estudio, se puede alcanzar el objetivo propuesto. Desde otro punto de vista, es posible acceder a soluciones de mayor calidad mediante el cambio de los parámetros que determinan la función objetivo, las características del problema y sus restricciones, además el ajuste de los parámetros de una estrategia de búsqueda, mejora los resultados obtenidos para un problema MFVRP determinado, sin que ello signifique que sea el mejor para otros escenarios.

Como se vio en la validación del modelo, la metodología propuesta en este trabajo conlleva una mejora sustancial en relación a los costos de transporte, representado en los vehículos empleados, el porcentaje de utilización de capacidad de cada camión y el más importante de todos, la distancia total recorrida. Por ejemplo para la salida del GTS con el método de barrido se tuvo un ahorro de 56.596,82 m en la distancia total recorrida, se evitó el uso de un vehículo y se mejoraron la distribución de clientes por ruta, evitando el pago de horas extras en una ruta. Para los resultados del GTS con el método de ahorros se disminuyó la distancia total en 8.654,24 m y se evitó el uso de un camión. Sin duda, la aplicación de esta metodología conlleva beneficios económicos representativos, si se llevaran los resultados obtenidos a sus equivalentes en pesos.

7.2 Trabajos Futuros

Futuras investigaciones podrían encaminarse a considerar los siguientes aspectos:

- ✓ Obtener por parte de la empresa los costos fijos y variables de cada vehículo con el fin de ingresarlos al algoritmo y obtener resultados más detallados.

- ✓ Cambiar la función objetivo para minimizar primero la cantidad de vehículos y luego la distancia recorrida. Algunas formas posibles de realizar esta tarea son: Agregar una penalización por cantidad de vehículos utilizados y Modificar y/o diseñar operaciones en función de buscar minimizar la cantidad de vehículos.

- ✓ Agregar otro conjunto de clientes a la lista de candidatos mediante otro filtro alternativo por cantidad de clientes en lugar de distancia. Para cada cliente se considera que otro cliente es vecino del primero si está dentro de los C clientes más cercanos sin importar la distancia que se encuentre. A esta variante de la Búsqueda Tabú se le conoce como Tabú Granular por Cantidad (Perosio & Zunino, 2008).

- ✓ También sería útil calcular los tiempos totales para cada ruta mediante el algoritmo y así poder desarrollar alguna estrategia con el fin de que cada ruta sólo emplee como máximo 8 horas en su trayecto diario. Este es un factor clave, ya que puede ayudar al modelo a ajustarse mucho más a la realidad.

- ✓ Se podría solucionar este caso estudio con otra muy buena metaheurística, como por ejemplo el recocido simulado, a fin de comparar la calidad de la solución y el tiempo empleado para obtener una respuesta.

- ✓ Como trabajo futuro también sería interesante someter la metaheurística diseñada a instancias más grandes, en otras palabras, aplicar el algoritmo a un problema con mayor cantidad de clientes y de vehículos, lo cual se asimilaría más a los casos encontrados en la práctica.

8. REFERENCIAS BIBLIOGRÁFICAS

Ahuja, R., Magnati, T., & Orlin, J. (1988). Network Flows. Cambridge, Mass.: Alfred P. Sloan School of Management. Massachusetts Institute of Technology.

Ahuja, R., Magnati, T., Orlin, J., & Reddy, M. (1994). *Applications of Network Optimization* (p. 83). Research Paper. Sloan School of Management. Massachusetts Institute of Technology.

Aponza, F., & Rodriguez, J. (2006). *Proceso de Logística Inversa del Reciclaje en la empresa IPP Ltda.* Tesis de Pregrado. Universidad Autónoma de Occidente.

Baldacci, R., Battarra, M., & Vigo, D. (2007). *Routing a Heterogeneous Fleet of Vehicles* (pp. 1–25). Technical Report. University Bologna. Venezia.

Ballou, R. H. (1999). *Logística. Administración de la Cadena de Suministro* (p. 816). México: Pearson Educación.

Belfiore, P., & Favero, L. (2007). Scatter search for the fleet size and mix vehicle routing problem with time windows. *Center European Journal of the Operational Research*, *15*(4), 351.

Berbotto, L., Garc, S., & Nogales, F. J. (2013). A Randomized Granular Tabu Search Heuristic for the Split Delivery Vehicle Routing Problem. *Annals of Operation Research*, *2*, 1–25.

Bräysy, O., Dullaert, W., Hasle, G., Mester, D., & Gendreau, M. (2006). An effective multi-restart deterministic annealing methauristic for the fleet size and mix vehicle routing problem with time windows. *Transportation Science*, *2*, 1–29.

Braysy, O., Porkka, P., Dullaert, W., Repoussis, P., & Tarantilis, C. D. (2009). A well-scalable metaheuristic for the fleet size and mix vehicle routing problem with time windows. *Expert Systems with Applications*, *36*(4), 8460–8475.

Clarke, G., & Wright, J. V. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, *12*, 568–581.

Cordeau, J. F., Laporte, G., Savelsbergh, M. W. P., & Vigo, D. (2007). Vehicle Routing. *Handbook in OR &MS*, *14*(06), 367–428. doi:10.1016/S0927-0507(06)14006-2

Cplusplus.com. (2013). The C++ Resources Network. Retrieved November 07, 2013, from http://www.cplusplus.com/

Crainic, T. G., Gendreau, M., Soriano, P., & Toulouse, M. (1993). A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements. *Annals of Operations Research*, *41*, 359–383.

Dantzig, G., Fulkerson, D., & Johnson, S. (1954). Solution of a large-scale traveling salesman problem. *Operations Research*, *1934*, 393–410.

Daza, J. M., Montoya, J., & Narducci, F. (2009). Resolución del problema de enrutamiento de vehículos con limitaciones de capacidad utilizando un procedimiento metaheurístico de dos fases. *Revista EIA*, *12*, 23–38.

Deitel, H., & Deitel, P. (2008). *Como Programar en C/C++y Java* (6ta ed., p. 1152). México: Pearson Educación.

Dell'Amico, M. M., Pagani, C., & Vigo, D. (2006). Heuristic approaches for the fleet size and mix vehicle routing problem with time windows. *Transportation Science*, *4*, 516–526.

Desrochers, M., & Verhoog, T. W. (1991). A new heuristic for the fleet size and mix vehicle routing problem. *Computers and Operations Research*, *18*, 263–274.

Díaz, A., Glover, F., Ghaziri, H. M., Gonzalez, J. L., Laguna, M., Moscato, P., & Tseng, F. T. (1996). *Optimización Heurística Y Redes Neuronales En Dirección De Operaciones E Ingeniería* (p. 235). Paraninfo.

Dullaert, W., Janssens, G. K., Sörensen, K., & Vernimmen, B. (2002). New heuristics for the fleet size and mix vehicle routing problem with time windows. *Operations Research*, *53*, 1232–1238.

Fisher, M., & Jaikumar, R. (1981). A generalized assignment heuristic for the vehicle routing problem. *Networks*, *11*(2), 109–124.

Gendreau, Hertz, A., & Laporte, G. (1992). New insertion and post optimization procedures for the traveling salesman problem. *Operations Research*, *40*, 1086–1094.

Gendreau, Hertz, A., & Laporte, G. (1994). Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, *40*(10), 1276–1290.

Gendreau, M. (2002). *An introduction to Tabu Search* (pp. 1–21). Research Paper. Université de Montréal. Montréal.

Gendreau, M., Laporte, G., Musaraganyi, C., & Taillard, É. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, *26*, 1153–1173.

Gendreau, M., Potvin, J. Y., Bräysy, O., Hasle, G., & Lokketangen, A. (2007). Metaheuristics for the Vehicle Routing Problem and its Extensions : A Categorized Bibliography. *CIRRELT*, *27*, 1–27.

Gheysens, F., Golden, B., & Assad, A. (1984). A comparison of techniques for solving the fleet size and mix vehicle routing problem. *OR Spektrum*, *6*, 207–216.

Gheysens, F., Golden, B., & Assad, A. (1986). A new heuristic for determining fleet size and composition. *Mathematical Programming Study*, *26*, 233–236.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, *13*(5), 533–549.

Glover, F., & Laguna, M. (2007). Tabu Search. In P. M. Pardalos, R. Graham, & D.-Z. Du (Eds.), *Handbook of Combinatorial Optimization* (2011th ed., pp. 1–94).

Glover, F., & Melián, B. (2003a). Búsqueda Tabú. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, *7*(19), 1–21.

Glover, F., & Melián, B. (2003b). *Introducción a la Búsqueda Tabú* (Vol. 03, pp. 1–36). Research Paper. Universidad de la Laguna. España.

Golden, B., Assad, S., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research*, *11*(1), 49–66.

Gómez, I. (2006). La logística y la ingeniería del marketing. Retrieved June 07, 2012, from http://igomeze.blogspot.com/

Hillier, F., & Lieberman, G. (2010). *Introducción a la Investigación de Operaciones* (Novena Edi., p. 1010). McGraw-Hill.

Johnson, D. S., & Mcgeoch, L. A. (1997). The Traveling Salesman Problem : A Case Study in Local Optimization. In E. H. Aarts & J. K. Lenstra (Eds.), *Local Search in Combinatorial Optimization* (pp. 215–310).

Joyanes A, L. (2003). *Fundamentos de programación. Algoritmos, Estructuras de datos y Objetos.* (3ra ed., p. 1012). McGraw-Hill.

Kotler, P. (1991). *Marketing Management* (p. 456). Boston: Prentice Hall, Inc.

Laguna, M. (1994). A Guide to implementing Tabu Search. *Investigación Operativa*, *4*(1), 1–21.

Lin, S.-W., Yu, V. F., & Chou, S.-Y. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, *38*(12), 1–15. doi:10.1016/j.eswa.2011.05.075

Liu, F.-H., & Shen, S. H.-Y. (1999). The fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research*, *50*(7), 721–732.

Liu, S., Haung, M., & Ma, H. (2009). An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review*, *45*, 434–445.

Marti, R. (2004). *Algoritmos Heurísticos en Optimización Combinatoria* (p. 31). Research Paper. Departament d'Estadística i Investigació Operativa.

Martí, R. (2003). *Procedimientos Metaheurísticos en Optimización Combinatoria* (Vol. 1, pp. 3–62). Research Paper. Departament d'Estadística i Investigació Operativa.

Mintransporte. (2011). *Diagnóstico del Sector Transporte Colombia 2011* (p. 112). Bogotá.

Mintransporte. (2012). *Transporte en Cifras 2011* (p. 92). Retrieved from http://www.mintransporte.gov.co/descargar.php?idFile=5302

Muhammad, A. K., Shahjalal, M., Faruque, F., & Hasan, I. (2011). Solving the Vehicle Routing Problem using Genetic Algorithm. *International Journal of Advanced Computer Science and Applications*, *2*(7), 126–131.

Ochi, L. S., Vianna, D. S., Drummond, L. M. A., & Victor, A. O. (1998). A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet. *Future Generation Computer Systems*, *14*(5-6), 285–292. doi:10.1016/S0167-739X(98)00034-X

Olivera, A. (2004). *Heurísticas para Problemas de Ruteo de Vehículos* (p. 63). Research Paper. Universidad de la República. Uruguay. Retrieved from aolivera@fing.edu.u

Or, I. (1976). *Traveling salesman-type combinatorial optimization problems and theeir relation to the logistics of regional blood banking*. Tesis Doctoral. North Western University.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, *41*, 421–451.

Osman, I. H. (2001). *Metaheuristics : A General Framework* (Vol. 13, pp. 169–170). Research Paper. School of Business and Center for Advanced Mathematical Sciences. Beirut.

Osman, I. H., & Kelly, J. P. (1996). *MetaHeuristics: theory and applications*. (I. H. Osman & J. P. Kelly, Eds.) (pp. 1–21). Boston: Kluwer Academic.

Osman, & Salhi, S. (1996). Local Search Strategies for the Vehicle Fleet Mix Problem. In *Modern Heuristic Search Methods* (pp. 131–153).

Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2011). An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem. *Journal of Heuristics*, *2*, 1–32. doi:10.1007/s10732-011-9186-y

Perosio, L., & Zunino, C. (2008). *Un Algoritmo Tabu Search Granular para el Problema de Ruteo de Vehículos con Ventanas de Tiempo y Entregas Parciales*. Tesis de Pregrado. Universidad de Buenos Aires.

Renaud, J., & Boctor, F. F. (2002). A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal Of Operational Research*, *140*, 618–628.

Repoussis, P., & Tarantilis, C. D. (2009). Solving the fleet size and mix vehicle routing problem with time windows via adaptive memory programming. *Transportation Research Part C: Emerging Technologies*, *8*(5), 695–712.

Riojas Cañari, A. C. (2005). *Conceptos, algoritmo y aplicación al problema de las N-Reinas*. Tesis de Pregrado. Universidad Mayor de San Marcos.

Rochat, Y., & Taillard, É. D. (1995). Probabilistic Diversification And Intensification In Local Search For Vehicle Routing. *Journal of Heuristics*, *1*, 147–167.

Sait, S. M., & Youssef, H. (1999). Iterative computer algorithms with applications in engineering: Solving combinatorial optimization problems. *Piscataway: IEEE Computer Society Press*, 1–248.

Salhi, S., & Rand, G. K. (1987). Improvement to vehicle routing heuristics. *Journal of the Operational Research Society*, *38*, 293–295.

Salhi, S., Sari, M., Saidi, D., & Touati, N. (1992). Adaptation of some vehicle fleet mix heuristics. *Omega 20*, *20*, 653–660.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, *33*(4), 894–909. doi:10.1016/j.cor.2004.08.002

Shen, S. H.-Y., & Liu, F.-H. (1999). A Method for Vehicle Routing Problem with Multiple Vehicle Types and Time Windows. *Proc. Natl. Sci. Counc. ROC(A)*, *23*(4), 526–536.

Solomon, M. M. (2010). Algorithms for and scheduling problems the vehicle routing with time window constraints. *INFORMS Journal on Computing*, *35*(2), 254–265.

Taillard, É. D. (1999). A heuristic column generation method for the heterogeneous fleet VRP. *Revue Française D'automatique, D'informatique et de Recherche Opérationnelle*, *3*(1), 1–14.

Tan, K. C., Lee, T. H., Chew, Y. H., & Lee, L. H. (2006). A Hybrid Multiobjective Evolutionary Algorithm For Solving Truck And Trailer Vehicle Routing Problems. *European Journal of Operational Research*, *172*, 855–885.

Toth, P. (2010). Slides - "Integer Linear Programming Refining Procedures for Vehicle Routing Problems." In *University of Bologna* (p. 51).

Toth, P., & Vigo, D. (2000). *The Vehicle Routing Problem* (p. 384). Philadelphia: SIAM Monographs on Discrete Mathematics and Applications.

Toth, P., & Vigo, D. (2003). The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *Informs Journal on Computing*, *15*(4), 333–346. doi:10.1287/ijoc.15.4.333.24890

Vélez, M. C., & Montoya, J. A. (2007). Metaheurísticos: Una alternativa para la solución de problemas combinatorios en administración de operaciones. *Revista EIA*, *8*, 99–115.

Vidal, C. J. (2010). *Planeación, Optimización y Administración de Cadenas de Abastecimiento.* (p. 642). Santiago de Cali: Escuela de Ingeniería Industrial y Estadística – Universidad del Valle.

Villegas, J. G., Lopera, D. C., & Sanchez, D. J. (2007). *Problemas de enrutamiento: variantes, aplicaciones y bibliografía básica*. Tesis de Pregrado. Universidad de Antioquia, Medellin.

Wassan, N. A., & Osman, I. H. (2002). Tabu search variants for the mix fleet vehicle routing problem. *Journal of the Operational Research Society*, *53*, 768–782.

Yepes, V. (2002). *Optimización heurística económica aplicada a las redes de transporte del tipo VRPTW*. Tesis Doctoral. Universidad Politécnica de Valencia.

Yworks. (2013). yEd - Graph Editor. Retrieved November 07, 2013, from http://www.yworks.com/en/products_yed_about.html

Zanjirani, R., Rezapour, S., & Kardar, L. (2011). *Logistics Operations and Management: Concepts and Models*. (Elsevier, Ed.) (First., p. 475). USA: Elsevier Inc.