UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS

TESIS

PROBLEMA DE RUTAS CON MÚLTIPLES DEPÓSITOS Y VENTANAS DE TIEMPO

PRESENTADA POR

SARAHÍ BERENICE BÁEZ VIEZCA

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:
MAESTRÍA EN CIENCIAS CON ORIENTACIÓN EN MATEMÁTICAS

DICIEMBRE 2014

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN.
FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS.

PROBLEMA DE RUTAS CON MÚLTIPLES DEPÓSITOS Y VENTANAS DE TIEMPO

Por

Sarahí Berenice Báez Viezca.

Como requisito parcial para obtener el grado de:

MAESTRIA EN CIENCIAS CON ORIENTACIÓN EN MATEMÁTICAS.

DICIEMBRE 2014

PROBLEMA DE RUTAS CON MÚLTIPLES DEPÓSITOS Y VENTANAS DE TIEMPO

Por

Sarahí Berenice Báez Viezca.

Como requisito parcial para obtener el grado:

MAESTRO EN CIENCIAS CON ORIENTACIÓN EN MATEMÁTICAS.

Miembros del comité:

Alvaro Eduardo Cordero Franco

_____

Dr.

Asesor

Iris Abril Martínez Salazar

_____

Dr.

Co-asesor

José Fernando Camacho Vallejo

_____

Dr.

Sinodal

Contenido

Contenido

Lista de figuras

Lista de tablas

AGRADECIMIENTOS

Agradezco primeramente a Dios, por darme la vida, las fuerzas, la capacidad y la ayuda para salir adelante, no sólo en estos dos años de maestría, si no estos 24 años de vida donde puedo decir que hasta aquí me ha ayudado el Señor y sé que lo seguirá haciendo.

En su palabra dice que: el temor a Jehová es el principio de la sabiduría, y el conocimiento del Santísimo la inteligencia (Proverbios 9:10), promesa que se ha cumplido en mi vida, pues Él se ha perfeccionado en mi debilidad y sin yo planearlo me ha llevado a cumplir metas que ni siquiera aspiraba. Gracias Dios porque TODO es por ti y para ti, en mis fuerzas no hubiera sido posible.

También quiero agradecer a mis papás por educarme y enseñarme a que no hay mejor vida más que tomada de la mano de Dios, e infundir en mí las ganas de superarme, así como nunca dejarme sola y apoyarme en TODO, por ser mis guías y consejeros en todo momento. Gracias también a mis hermanos que junto con mis papás han aguantado todas mis crisis de estrés y darme la mano siempre que lo necesito. A mi novio Heber que también me ha aguantado bastante (jeje pobrecito) y nunca me ha dejado sola, gracias porque siempre me puedo desahogar contigo, y no necesito pedirte ayuda, porque antes de que te la pida tú ya me la estás dando, gracias Heber por quererme y aguantarme tanto.

También agradezco a mi asesor el Dr. Cordero, por su guianza y apoyo para entrar a la maestría y para salir de ella, por sus enseñanzas y por la paciencia con la que me asesoró estos dos años, sin dejar nunca de lado su buen trato con calidad humana y empatía para con sus alumnos.

Gracias a mis amigos y compañeros de maestría: Pana, Cristina, Trini y Tenorio, la maestría no hubiera sido la misma sin ustedes para hacerla mucho más amena; a Edna mi compañera de sufrimiento en las materias de análisis funcional y de análisis numérico, sólo ella compartía el mismo grado de felicidad cuando terminaban esas clases; a Norma, Selene, Diego, Jonathan por sus consejos y buenas pláticas.

Resumen

En la cadena de suministro intervienen diversos componentes que actúan para lograr que el producto llegue a las manos del cliente en condiciones óptimas; uno de estos factores es la distribución de los productos de los depósitos (almacenes) a los clientes (o minoristas), buscando que dicho proceso se realice con un costo mínimo y cumpliendo ciertas especificaciones para la entrega del producto a los clientes. Es por ello que el problema de rutas con múltiples depósitos (Multi-Depot Vehicle Routing Problem, MDVRP) busca satisfacer la demanda de todos los clientes minimizando el costo total de las rutas que parten de distintos depósitos para entregar los productos a los clientes, más aún si los clientes especifican horarios para la recepción de los productos, éste problema se conoce como un problema de rutas con múltiples depósitos y ventanas de tiempo (Multi-Depot Vehicle Routing Problem with Time Windows, MDVRPTW).

Este problema, es clasificado como NP-hard debido al gran costo computacional que se requeriría para encontrar una solución óptima de manera exhaustiva. Es por ello que en este trabajo se propone abordar tal problema con un algoritmo constructivo basado en la idea de rutear primero y agrupar después. Posteriormente se aplicará una etapa de mejora a la solución inicial dada en la primera etapa, buscando con esto encontrar soluciones cercanas al óptimo para este problema.

CAPÍTULO 1. INTRODUCCIÓN

1.1. Introducción

La cadena de suministro es la entrega al cliente de valor económico por medio de la administración sincronizada del flujo físico de bienes con información asociada de las fuentes de consumo (LaLonde, 1994). Ésta se compone (generalmente) de los siguientes eslabones: proveedor, fabricante, distribuidor y cliente.

En muchos de los casos los gastos de producción, distribución, almacenamiento, entre otros, tienden a ser muy elevados, lo que repercute drásticamente en el precio del producto por lo que es necesario minimizar dichos costos sin afectar la calidad del producto y los tiempos involucrados en los respectivos procesos; todo esto cumpliendo con los requerimientos que exigen cada una de las partes involucradas en la cadena de suministro.

En la actualidad la distribución de bienes o servicios por parte de una empresa a varios clientes o dependencias de la misma ha adquirido relevancia debido a los costos y especificaciones concernientes a su transportación, ya que es necesario crecer y mejorar en los servicios de entrega a medida que una empresa aumenta su mercado de clientes, pues cuando los costos de transportación y entrega de los bienes o servicios se acercan o exceden a las ganancias, el negocio ya no es redituable.

Se han hecho diversos estudios referentes a este tema en el área de optimización teniendo como objetivo la búsqueda de métodos para mejorar la distribución de productos por parte de las empresas.

Entre otros problemas, se ha trabajado con la búsqueda de las rutas por las cuales se transportarán los productos que minimicen el costo (o tiempo) de transporte. Éstas rutas se definen por las ubicaciones de los clientes con respecto al centro de distribución, el

número de vehículos disponibles, las capacidades de los mismos, tiempos de entrega, entre otras especificaciones.

En este trabajo se emplean dos heurísticas constructivas y diversos métodos de mejora, para resolver un problema de rutas de vehículos con múltiples depósitos y ventanas de tiempo (MDVRPTW, por sus siglas en inglés), donde el término de ventanas de tiempo se refiere a rangos de horarios de entrega especificados por los clientes, que según la naturaleza del problema, su cumplimiento debe de ser riguroso o en algunos casos solamente es deseable o preferente su cumplimiento.

En la siguiente sección de este capítulo se habla brevemente de la historia del MDVRPTW y sus antecedentes, en las demás secciones se tratan las preguntas de investigación que dan pie a este trabajo, así como la hipótesis propuesta, justificación y finalmente los alcances y limitaciones para esta investigación.

1.2. Historia y Antecedentes

El problema de rutas fue tratado por primera vez en el área de optimización, por Dantzig et al. (1959) como una generalización del problema del agente viajero (TSP, por sus siglas en inglés), el cual consiste en encontrar una ruta para un vehículo, que pase por n ciudades una sola vez sin subtours, desde una ciudad origen hasta una ciudad destino (puede ser la misma ciudad de origen) minimizando la suma de los costos de pasar por cada arco, o bien, la suma de sus distancias.

Este problema del agente viajero es la base del problema de rutas de vehículos (VRP, por sus siglas en inglés), el cual ha sido adaptado para cumplir con diversas necesidades para problemas reales de ruteo en la cadena de suministro. Entre las variantes de VRP se encuentra el VRP capacitado en el cual se agregan restricciones de capacidad de diversos tipos, por ejemplo la capacidad de carga de los vehículos. Otra variante corresponde al VRP con múltiples depósitos (MDVRP, por sus siglas en inglés) donde se cuenta con más de un depósito para enviar los productos a los clientes y satisfacer sus demandas y la flota de

vehículos puede estar disponible desde un depósito o un número fijo de vehículos disponibles para todos los depósitos en total. Diversos modelos de VRP han sido considerados, en los cuales se añaden distintas especificaciones como las características de la flota de vehículos, ya sea homogénea o heterogénea; el tipo de demanda de los clientes, determinística (previamente conocida como uno de los parámetros) o estocástica (la demanda de cada cliente es desconocida al iniciar la jornada); el tipo de servicio, ya sea entrega o recepción de los productos o ambos; entre muchas otras. Todas estas variantes del VRP han surgido con el fin de adaptar el problema de rutas de vehículos cada vez mejor a los problemas de distribución de productos o servicios de la vida real, es decir, entre más especificaciones o restricciones se la agregan al VRP, éste pasa de ser de un problema general de ruteo a un problema particular de rutas de vehículos.

Una variante más del VRP surgió al añadir ventanas de tiempo al problema de ruteo de vehículos para el caso de un solo depósito, esto fue propuesto por Solomon (1983), quien además proporcionó un conjunto de instancias para el VRPTW conocidas como "Instancias de Solomon", las cuales se convirtieron en casos de referencia para el VRPTW (Vehicle Routing Problem with Time Windows).

Otro caso resulta de combinar dos variantes del VRP, como el tener ventanas de tiempo dadas por los clientes en un problema de rutas de vehículos con múltiples depósitos, tornándose en un Problema de Rutas de Vehículos con Múltiples Depósitos y Ventanas de Tiempo (MDVRPTW, por sus siglas en inglés) ya mencionado anteriormente.

1.3. Declaración del problema

El problema a tratar en esta tesis es un problema de rutas de vehículos con múltiples depósitos y ventanas de tiempo. El objetivo del modelo es minimizar los costos de distribución de los productos; partiendo de L depósitos y contando con una flota de S vehículos homogéneos en cada depósito para transportar los productos a R clientes que especifican los rangos de horarios en que recibirán los productos. Estos rangos de horarios

proporcionados por los clientes, se conocen como *ventanas de tiempo* y se clasifican en ventanas de tiempo suaves y estrictas.

Nosotros vamos a considerar ventanas de tiempos suaves, es decir, se permite que un vehículo arribe antes o después de la ventana de tiempo del cliente, incurriendo en una penalización en la función objetivo; mientras que las ventanas de tiempo estrictas, son aquellas donde no se permite el incumplimiento en los horarios de entrega especificados por los clientes.

Para resolver este problema se trabaja con un algoritmo compuesto de dos etapas, la primera etapa consiste en la construcción de una solución inicial, y la segunda etapa es de mejora, en la que con movimientos de búsqueda local se pretende mejorar la solución inicial encontrada en la primera etapa.

Para la primera etapa, se propone una heurística constructiva, basada en la estrategia de la inserción más barata para crear una mega ruta que contenga a todos los clientes asignados para cada depósito y posteriormente hacer cortes en ella para cumplir con las restricciones de capacidad de carga y duración máxima permitida para la ruta.

1.4. Preguntas de Investigación

Pregunta 1.- La construcción de una solución inicial para el problema de rutas de vehículos con múltiples depósitos y ventanas de tiempo generando una mega ruta a partir de un cliente semilla y el procedimiento de inserción en base a ahorros, ¿generará soluciones iníciales factibles y de buena calidad?

Pregunta 2.- La implementación de una etapa de mejora para el problema de rutas con múltiples depósitos y ventanas de tiempo, ¿permitirá encontrar rápidamente mejores soluciones para el problema de rutas de vehículos con múltiples depósitos y ventanas de

tiempo, dado que por la naturaleza de dicho problema, éste cuenta con pocas y aisladas soluciones factibles?

1.5. Hipótesis de Investigación

Hipótesis 1.- La metodología de primero construir una mega ruta y después agrupar, puede generar buenas soluciones iníciales factibles.

Hipótesis 2.- Los movimientos de búsqueda local que se implementarán en la segunda etapa, mostrarán un buen desempeño en la búsqueda de soluciones factibles.

1.6. Objetivo

Implementar un algoritmo heurístico que sea eficiente tanto en la construcción de una solución inicial factible, como en la búsqueda de mejores soluciones factibles que estén cercanas al óptimo o que lo encuentren, para el problema a tratar en esta tesis.

1.7. Justificación

Una de las principales complicaciones que surgen al intentar resolver el problema descrito en la sección anterior son las restricciones de ventanas de tiempo, debido a que se busca satisfacer al cliente no sólo con el cumplimiento de los pedidos, sino también con el cumplimiento de los horarios dados por ellos para recibir los pedidos; un ejemplo de ello son los laboratorios farmacéuticos especializados que además de surtir a farmacias también surten a pacientes con enfermedades que requieren medicamento especial y cuentan con una membrecía; en muchos de los casos el paciente llama para pedir el medicamento e indica el horario en que lo puede recibir y el medicamento es entregado en

ese rango de tiempo, esta es solo una de múltiples aplicaciones reales que involucran las ventanas de tiempo; realzando de esta forma la importancia de analizar este tipo de problemas.

1.7.1. Justificación Científica

Como ya se dijo, el problema de rutas con múltiples depósitos y ventanas de tiempo, tiene como dificultad particular el cumplimiento de las restricciones de ventanas de tiempo, las cuales provocan que las soluciones factibles sean escasas y aisladas entre sí.

Hasta ahora, la mayoría de los trabajos encontrados en la literatura que trabajan con ventanas de tiempo son para problemas de rutas con un solo depósito, por lo que este trabajo trata de dar una aportación más a los trabajos hechos para varios depósitos.

1.7.2. Justificación Práctica

En la vida real son comunes los problemas de transporte o distribución de bienes, la particularidad de ventanas de tiempo dadas por los clientes está presente no solo en la industria farmacéutica, también se trabaja de esta manera en ciertas cadenas de tiendas de conveniencia donde los camiones que surten de mercancía a las mismas solamente son recibidos en horario nocturno cuando éstas tiendas cierran o tienen menor afluencia de clientes y de esta manera no se crea conflicto entre la atención al cliente y la recepción de los bienes.

1.8. Alcances y limitaciones

Cómo ya se mencionó el VRP es un problema muy general que al agregarle especificaciones a los elementos que lo componen se vuelve un problema cada vez más real y aplicable en la industria; sin embargo, entre más restricciones se consideren el problema se vuelve más intratable computacionalmente y aunado a que el VRP es un problema NP-hard, encontrar la solución óptima se vuelve cada vez más complicado.

En este trabajo el problema a tratar presenta mucha dificultad para encontrar soluciones factibles, debido a que las ventanas de tiempo dadas por los clientes influyen fuertemente en la construcción de las rutas. Una de las principales razones es que ahora no sólo se consideran las restricciones de capacidad y de tiempo máximo de duración de la ruta, sino también que cada cliente sea atendido a la hora que especificó.

Considerando esto, uno de los alcances de este trabajo, es que las ventanas de tiempo a tratar (como ya se mencionó anteriormente) serán restricciones suaves, es decir, el cliente puede ser atendido antes o después del rango de hora acordado, pero se aplicará una penalización o multa en la función objetivo que aumente los costos de distribución, con el fin de que aunque se permite incumplir con la ventana de tiempo, esto no sea lo ideal, ya que si se consideraran ventanas de tiempo estrictas el conjunto de soluciones factibles sería más pequeño y soluciones factibles estarían más aisladas entre sí.

Una limitación es que en este trabajo se considera un número fijo de vehículos para cada uno de los depósitos y éstos sólo pueden hacer una ruta, es decir, sólo se permite una ruta por vehículo; además que cada cliente es visitado una sola vez, por lo que la demanda de cada cliente debe ser satisfecha en su totalidad por el vehículo asignado a éste.

CAPÍTULO 2. MARCO TEÓRICO

En este trabajo se aborda la variante del problema de rutas de vehículos con ventanas de tiempo y múltiples depósitos (MDVRPTW), por lo que la primera sección de este capítulo se dedica al VRP en general y la segunda al MDVRPTW, para dar pie a hablar de algunas heurísticas con las que se ha trabajado el VRP y algunas de sus extensiones.

2.1 Problema de rutas de vehículos (VRP).

El problema de rutas de vehículos (VRP, por sus siglas en inglés) ha sido extensamente estudiado desde que Dantzig et. al. (1959) lo introdujo en su artículo "The truck dispatching problem", donde trató el problema de rutas de vehículos para una flota de camiones para el reparto de gasolina en distintas estaciones de servicio con una terminal o estación centro de la cual parten los camiones; éste problema lo abordó como una extensión del problema del agente viajero (TSP, por sus siglas en inglés), el cual consiste en encontrar una ruta para un vehículo, que pase por n ciudades una sola vez sin subtours, desde una ciudad origen hasta una ciudad destino (puede ser la misma ciudad de origen) minimizando la suma de los costos de pasar por cada arco (el costo puede estar en los nodos), o bien, la suma de sus distancias.

Debido a su complejidad, el problema del agente viajero es considerado como NP-hard, por lo que todas las variantes del TSP, esto incluye el VRP y sus variantes, son NP-hard también.

Este problema del agente viajero es la base del problema de rutas de vehículos (VRP), que con el tiempo se ha diversificado conforme se le han agregado restricciones o bien se han variado ciertas características en los factores que intervienen en éste problema. Buscando con esto adaptar el problema de ruteo de vehículos para cubrir más ampliamente las necesidades de los problemas de distribución en la vida real y poder dar resultados más eficientes y útiles en la industria.

A continuación se presenta un modelo para el VRP, usando una variable binaria x_{ij} que toma el valor de 1 si se utiliza el arco (i,j) ϵ A y toma el valor de 0 en otro caso; donde A es el conjunto de arcos que pasan por los clientes y el depósito, y K es el número de vehículos disponible, en este caso todos los vehículos deben utilizarse y habrá una ruta por vehículo, esto es, $|k|$ rutas.

$$(2.1) \quad \min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

$$(2.2) \quad \sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V - \{0\},$$

$$(2.3) \quad \sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V - \{0\},$$

$$(2.4) \quad \sum_{i \in V} x_{i0} = k$$

$$(2.5) \quad \sum_{j \in V} x_{0j} = k$$

$$(2.6) \quad \sum_{i \notin S} \sum_{j \notin S} x_{ij} \geq r(S) \qquad \forall S \subseteq V - \{0\},\ S \neq \{\ \}$$

$$(2.7) \quad x_{ij} \in \{0,1\} \quad \forall i, j \in V.$$

Donde las restricciones (2.2) y (2.3) implican que solamente se activa un arco para llegar al cliente y solamente uno para salir de él, con esto aseguramos también que cada cliente es atendido una sola vez; las restricciones (2.4) y (2.5) implican que del depósito salen k vehículos y regresan k vehículos; la restricción (2.6) impide que una solución tenga subtours, y la restricción (2.7) muestra la naturaleza de las variables.

La primera variante que encontramos en la literatura es el VRP capacitado (CVRP, por sus siglas en inglés) dónde se consideran restricciones de capacidad en los vehículos y/o duración máxima de las rutas de cada vehículo, aunadas a las restricciones clásicas del VRP que son: cada cliente es visitado una sola vez por un solo vehículo, cada ruta inicia y termina en el depósito y la restricción de no subtours; ahora, si todos los vehículos tienen

la misma capacidad se dice que es una flota homogénea, de lo contrario es entonces una flota de vehículos heterogénea y las capacidades de las rutas varían.

Para el caso en que tenemos más de un depósito para hacer la distribución, entonces el VRP se convierte en un MDVRP, para el cuál las especificaciones sobre los depósitos son diversas, el caso más usual es en el que las rutas salen y regresan al mismo depósito y no se permite que en la ruta exista otro depósito. Algunos que han trabajado con este problema son Renaud et. al. (1996) con una flota homogénea de vehículos y poniendo la condición de que todos los depósitos deben de ser utilizados, y Surekha y Sumathi (2011) quienes trabajaron con un algoritmo genético para resolverlo; otro caso es cuando las rutas si pueden pasar por más de un depósito distinto, esta variante se conoce como problema de rutas de vehículos entre múltiples depósitos (MDVRPI, por sus siglas en inglés), trabajado por Crevier et al. (2007) con una formulación especial, ya que, para el conjunto de rutas hechas por un mismo vehículo se le nombra rotación y maneja dos tipos de rutas, una es cuando la ruta inicia y termina en el mismo depósito, esto es, ruta de un sólo depósito, y el segundo tipo de ruta es cuando la ruta conecta 2 depósitos diferentes, esto es, ruta entre depósitos.

Otro tipo de variante del VRP es cuando el costo de la ruta también depende de la carga, y esto se refleja en la función objetivo que también comprende los costos relacionados con las distancias recorridas por las rutas y el uso de los vehículos, Fung et Al. (2009) trabajó con el problema de rutas de vehículos con múltiples depósitos con costos relacionados a los pesos (MDVRPWRC, por sus siglas en inglés) incluyendo la carga del vehículo como una variable en la función objetivo. El MDVRPWRC busca determinar la secuencia para visitar los clientes en las rutas con el fin de minimizar los costos ya mencionados.

Cuando los vehículos no sólo entregan productos o bienes, sino que también recogen bienes por parte de los clientes, entonces estamos hablando de un problema de rutas de vehículos con entrega y recepción de productos o bienes simultáneamente (VRPSPD, por sus siglas en inglés), un ejemplo práctico para este caso es el dado por Min, H. (1989) de

una biblioteca pública que entrega y recoge los libros en préstamo, introduciendo así, dicho problema. Otro trabajo relacionado es el de Dethloff, J. (2001) que da como ejemplo el caso de la industria de bebidas, que como es bien sabido, no sólo requieren entregar la carga de botellas de vidrio para vender a los minoristas, si no también recoger las botellas vacías que los consumidores regresan a las tiendas para poder comprar la bebida nuevamente, y marca la diferencia entre el VRPSPD y el problema de ruteo de vehículos con regresos o viajes de retorno (VRPB, pos sus siglas en inglés). En este problema se tienen dos conjuntos de clientes, uno es el de los clientes que sólo reciben los productos y otro es el de los clientes que sólo entregan productos o bienes para que los vehículos los lleven al depósito. En este caso cada vehículo atiende primero a los clientes a los que va a entregar los productos y después al conjunto de clientes que le entregarán los productos o bienes.

Ahora bien, si las demandas de los clientes se pueden satisfacer en diferentes días, entonces es necesario planificar las rutas dadas para cada día y cuales clientes serán visitados en cada uno de estos días por dichas rutas, por lo que se trata de un problema de planeación rutas de vehículos (PVRP, por sus siglas en inglés). Éste es una extensión también del problema del agente viajero por periodos (PTSP, por sus siglas en inglés) y ambos problemas fueron trabajados por Cordeau et. al. (1997), junto con el MDVRP usando una misma formulación para los tres, a partir de la formulación dada para el PVRP; si en lugar de t número de días toman $t=1$ entonces esa misma formulación sirve para el VRP, ahora cuando el número de vehículos m es igual a 1 y la capacidad de carga del vehículo junto con su tiempo máximo de trabajo son igual a infinito, entonces la formulación se convierte en una formulación para el PTSP, y para que esa misma formulación original pase a ser un formulación para el MDVRP, sólo es necesario considerar los t días como t depósitos; además con esta formulación también se puede trabajar con una flota de vehículos tanto heterogénea como homogénea; los tres problemas fueron resueltos con un algoritmo de búsqueda tabú.

A continuación se presenta dicho modelo, para el cual se definió un grafo $G=(V,A)$, dónde el conjunto de vértices es $V=\{v_0,v_1,...,v_n\}$ y el conjunto de arcos es $A=\{(v_i,v_j)^{k,l}\}$, los superíndices k y l indican el número de vehículo y el número de día en que se hace la visita, respectivamente. Los costos de transporte de ir de un cliente i a un cliente j con el vehículo k en el día l se indican como c_{ijkl} . Las variables con las que trabaja el modelo son: las variables binarias x_{ijkl} igual a 1 si y sólo si el vehículo k visita al cliente j inmediatamente después de haber visitado al cliente i ($i \neq j$) el día l e igual a 0 en otro caso, y y_{ir} igual a 1 si sólo si la combinación de visitas $r \in C_i$ es asignada al cliente i; también se definió una constante binaria para el PVRP, a_{rl} igual a 1 si y sólo si el día l pertenece a la combinación r.

(2.8) $$\min \sum_{i=0}\sum_{j=0}\sum_{k=1}\sum_{l=1} c_{ijkl}\, x_{ijkl}$$

(2.9) $$\sum_{r\in C_i} y_{ir} = 1 \qquad \forall i = 1:n,$$

(2.10) $$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{ijkl} - \sum_{r\in C_i} a_{rl}\, y_{ir} = 0 \qquad \forall i = 1:n, \forall l = 1:t,$$

(2.11) $$\sum_{i=0}^{n} x_{ihkl} - \sum_{j=0}^{n} x_{hjkl} = 0 \qquad \forall h = 1:n, \forall k = 1:m, \forall l = 1:t,$$

(2.12) $$\sum_{j=1}^{n} x_{0jkl} \leq 1 \qquad \forall k = 1:m, \forall l = 1:t,$$

(2.13) $$\sum_{i=0}^{n}\sum_{j=0}^{m} q_i\, x_{ijkl} \leq Q_k \qquad \forall k = 1:m, \forall l = 1:t,$$

(2.14) $$\sum_{i=0}^{n}\sum_{j=0}^{n} (c_{ijkl} + d_i)\, x_{ijkl} \leq D_k \qquad \forall k = 1:m, \forall l = 1:t,$$

(2.15) $$\sum_{v_i\in S}\sum_{v_j\in S} x_{ijkl} \leq |S| - 1 \qquad \forall k = 1:m, \forall l = 1:t, S \subseteq (V - \{0\}), |S| \geq 2,$$

(2.16) $$x_{ijkl} \in \{0,1\} \qquad \forall i = 0:n, \forall j = 0:n, \ \forall k = 1:m, \forall l = 1:t,$$

(2.17) $$y_{ir} \in \{0,1\} \qquad \forall i = 0:n, \forall r \in C_i.$$

La restricción (2.9) asigna a cada cliente a cada una combinación factible, la restricción (2.10) garantiza que cada cliente es visitado solamente los días correspondientes a la combinación Ci asignada, mientras que la restricción (2.11) lo que hace es garantizar que si

un vehículo llega a un cliente en un día, ese mismo día sale; la restricción (2.12) indica que cada vehículo es usado a lo más una vez al día. La capacidad de carga y duración máxima de los vehículos son restringidas con las restricciones (2.13) y (2.14) respectivamente; para eliminar los subtours se utiliza la restricción (2.15) y las restricciones (2.16) y (2.17) indican la naturaleza binaria de las variables.

Así mismo Solomon. M., (1983) propuso una variante más para el VRP, al introducir el concepto de ventanas de tiempo (TW, por sus siglas en inglés) y crear la llamadas "Instancias de Solomon", un conjunto de instancias donde todas las distancias son euclideanas y la velocidad de los vehículos es la unidad, es decir, el vehículo avanza una unidad de distancia en una unidad de tiempo lo que permite tratar con los costos de trayecto c_{ij}, los tiempos de trayecto t_{ij} y las distancias entre clientes y depósitos de manera equivalente; dichas instancias fueron creadas específicamente para problemas con ventanas de tiempo.

A partir de entonces este nuevo problema ha sido extensamente trabajado, incluso con otras variantes del VRP, como en el caso de Paraskevopoulos, D., et. al. (2008) donde trataron el VRPTW con una flota heterogénea de vehículos; otro ejemplo es el trabajo de Potvin, J. y Rosseau, J. (1993) que trabajaron el Problema de Rutas de Vehículos y Programación con Ventanas de Tiempo (VRSPTW, por sus siglas en inglés).

A continuación se presenta un modelo para el VRPTW presentado en Toth, P., y Vigo, D., (2001), donde las ventanas de tiempo de los clientes se tomaron como estrictas y el nodo depósito es representado con el 0 para cuando los *K* vehículos parten de él y como *n+1* para cuando regresan, además se considera una flota de vehículos homogénea. En este modelo se sigue una notación parecida a la del modelo presentado para el VRP anteriormente, donde el conjunto $N = V\backslash\{0, n+1\}$ representa el conjunto de clientes; la variable binaria x_{ijk} es igual a 1 si el vehículo *k* va del cliente *i* al cliente *j* e igual a 0 en otro caso; la variable continua w_{ik} especifica el tiempo en que el vehículo *k* inicia el servicio para el cliente *i*.

$$(2.18)\quad \min \sum_{i\in V}\sum_{j\in V} c_{ij}\, x_{ijk}$$

$$(2.19)\quad \sum_{k\in K}\sum_{i\in V} x_{ijk} = 1 \qquad \forall i \in N,$$

$$(2.20)\quad \sum_{j\in N+\{0\}} x_{0jk} = 1 \qquad \forall k \in K,$$

$$(2.21)\quad \sum_{\substack{i\in N\\ i\neq j}} x_{ijk} - \sum_{\substack{i\in N\\ i\neq j}} x_{jik} = 0 \qquad \forall k \in K, \forall j \in N,$$

$$(2.22)\quad \sum_{i\in N+\{0,n+1\}} x_{i,n+1,k} = 1 \qquad \forall k \in K,$$

$$(2.23)\quad x_{ijk}(w_{ik} + s_i + t_{ij} - w_{jk}) \leq 0 \qquad \forall k \in K, \forall (i,j) \in A,$$

$$(2.24)\quad a_i \sum_{j\in N} x_{ijk} \leq w_{ik} \leq b_i \sum_{j\in N} x_{ijk} \qquad \forall k \in K, \forall i \in N,$$

$$(2.25)\quad E \leq w_{ik} \leq L \qquad \forall k \in K, \forall i \in \{0, n+1\},$$

$$(2.26)\quad \sum_{i\in N} d_i \sum_{\mathrm{j}\in \mathrm{N}} x_{ijk} \leq C \qquad \forall k \in K,$$

$$(2.27)\quad x_{ijk} \geq 0 \qquad \forall k \in K, \forall (i,j) \in A,$$

$$(2.28)\quad x_{ij} \in \{0,1\} \quad \forall k \in K, \forall (i,j) \in A, \quad w_{ik} \in \mathrm{R}.$$

La restricción (2.19) asigna sólo un cliente para cada vehículo, las restricciones (2.20) y (2.21) especifican que el vehículo *k* sale del depósito solo hacia un cliente y sólo regresa a partir de un cliente, asimismo la restricción (2.22) es también restricción de flujo e implica que el vehículo que llega al cliente j también sale, la restricción (2.23) controla los tiempos de arribo de los vehículos a los clientes de manera que cuadren, la restricción (2.24) es la restricción para el cumplimiento de las ventanas de tiempo de los clientes, así como la (2.25) para la ventana de tiempo dada para el depósito, ya que $a_0 = E \leq min_{i\in N}(b_i - t_{0i})$ y $b_{n+1} = L \leq min_{i\in N}(a_i + s_i + t_{ij})$, la restricción (2.26) es para la capacidad de los vehículos y las restricciones (2.27) y (2.28) indican la naturaleza de las variables.

Para el VRPTW las ventanas de tiempo también se pueden considerar como suaves, las cuales se explican brevemente en Kallehauge, B. et. al. (2006) que permiten el incumplimiento en la ventana de tiempo, pero se penaliza en la función objetivo, de forma proporcional al tiempo de incumplimiento.

Esta restricción es considerada tanto para el VRP como para el MDVRP, pero la trataremos más a fondo en la segunda sección de este capítulo, ya que está presente en nuestro problema de investigación.

2.2 Problema de ruteo de vehículos con ventanas de tiempo y múltiples depósitos (MDVRPTW).

En la sección anterior se estableció el VRP en general con algunas de sus variantes con más aplicaciones, como introducción a la variante estudiada en esta investigación, el MDVRPTW.

El VRPTW consiste en obtener un conjunto de rutas que parten de diferentes depósitos y satisfagan la demanda de todos los clientes y además los atiendan dentro de las ventanas de tiempo dadas por ellos, minimizando el costo de viaje.

Como ya se dijo en el capítulo anterior, en este caso se consideran ventanas de tiempo suaves, y se abordan de la siguiente manera, si el vehículo s llegó en un tiempo t_i^s al cliente i y su ventana de tiempo inicia en el tiempo e_i , si t_i^s es menor que e_i, entonces la multa P_i^1 a cobrar sería $P_i^1(e_i - t_i^s)$ y si $l_i$es el tiempo final de la ventana de tiempo del mismo cliente i, y si t_i^ses mayor que l_i entonces la multa P_i^2 a cobrar es $P_i^2(l_i - t_i^s)$; buscando con esto, evitar que se incumplan las dichas ventanas de tiempo.

Otros supuestos que se asumen durante esta investigación son que: la flota de vehículos es homogénea, los depósitos tendrán la misma capacidad en cuanto a las rutas, solo se puede atender al cliente asignado a la ruta una sola vez y satisfacer su demanda completa, y cada

cliente es atendido por solo una ruta, donde un vehículo hace solamente una ruta, la cual no excederá el tiempo máximo permitido para la ruta (tiempo máximo para la jornada).

A continuación se presenta la formulación dada en Dondo, R., & Cerdá, J. (2007) para el MDVRPTW, en el cual las ventanas de tiempo se consideran estrictas, sin embargo si un vehículo llega antes del tiempo de inicio de la ventana de tiempo del cliente a_i para el clientes i, entonces se cobra el tiempo Δa_i que el vehículo debe de esperar a que inicie la ventana de tiempo de dicho cliente, en cambio se prohíbe que el vehículo arribe al cliente después del cierre de la ventana de tiempo del cliente, entonces $\Delta b_i = 0$.

Para este modelo se define el conjunto de clientes I, el conjunto de depósitos P y el conjunto de vehículos V. También se define la matriz de costos dependientes de las distancias entre nodos c_{ij}^v y el menor tiempo de trayecto t_{ij}^v de ir del nodo i al nodo j. Para los vehículos se definen los parámetros de capacidad q_v del vehículo, duración máxima de la ruta r_v para el vehículo v, los costos fijos cf_v por uso del vehículo v y el máximo tiempo de trabajo permitido tv_v^{max} para el vehículo v. Y para los clientes se tienen: la demanda w_i de cada cliente, el tiempo de servicio st_i^v para el cliente i por el vehículo v y las ventanas de tiempo de cada cliente $[a_i, b_i]$; además, asociados a las ventanas de tiempo de los clientes se tienen los costos de penalización ρ_i por incumplimiento de las ventanas de tiempo y también los costos de penalización ρ_v por unidad de tiempo por incumplimiento del máximo tiempo de trabajo permitido para el vehículo v.

Y se trabaja con las siguientes variables binarias: $X_{\rho v}$ igual a 1 si el vehículo v es asignado al depósito ρ y cero en otro caso, Y_{iv} igual a 1 si el cliente i es asignado al vehículo v y cero en otro caso, y para seguir el orden la variable s_{ij} igual a 1 si el nodo i es visitado antes del nodo j. Además se cuenta con las siguientes variables continuas: Δa_i tiempo durante el cual se incumplió la ventana de tiempo del cliente i debido a que el vehículo llegó antes de su inicio, Δb_i tiempo durante el cual se incumplió la ventana de tiempo del cliente i debido a que el vehículo llegó después de su cierre, ΔT_v tiempo de incumplimiento del máximo tiempo de trabajo permitido para el vehículo v. Las variables relacionadas a los costos de las rutas son: C_i costo de la distancia acumulada al llegar al nodo i y CV_v costo de la

distancia total recorrida por el vehículo v. De manera análoga se tienen las siguientes variables en cuanto a tiempo: T_i tiempo en que el vehículo llega al cliente i y TV_v duración de la ruta hecha por el vehículo v.

(2.29) $$\min \quad \sum_{v\in V}\left(cf_v\sum_{p\in P}x_{pv}+c_t TV_v+CV_v\right)+\rho_v\Delta T_v+\sum_{i\in I}\rho_{i_i^1}(\Delta a_i+\Delta b_i)$$

Sujeto a :

(2.30) $$\sum_v Y_{iv}=1 \qquad \forall i\in I$$

(2.31) $$\sum_p X_{pv}\le 1 \qquad \forall v\in V$$

(2.32) $$c_i \ge c^v_{pi}(X_{pv}+Y_{iv}-1) \qquad \forall i\in I,\forall p\in P,\forall v\in V$$

(2.33) $$c_j \ge c_i + c^v_{ij} - M_C(1-s_{ij}) - M_C(2-Y_{iv}-Y_{jv}) \qquad \forall i\in I,\forall p\in P,\forall v\in V$$

(2.34) $$c_i \ge c_j + c^v_{ji} - M_C s_{ij} - M_C(2-Y_{iv}-Y_{jv}) \qquad \forall i\in I,\forall p\in P,\forall v\in V$$

(2.35) $$CV_v \ge C_i + c^v_{ip} - M_c(2-X_{pv}-Y_{iv}) \qquad \forall i\in I,\forall p\in P,\forall v\in V$$

(2.36) $$T_i \ge t^v_{pi}(X_{pv}+Y_{iv}-1) \qquad \forall i\in I,\forall p\in P,\forall v\in V$$

(2.37) $$T_j \ge T_i + st_i + t^v_{ij} - M_T(1-s_{ij}) - M(2-Y_{iv}-Y_{jv}) \qquad \forall i\in I,\forall p\in P,\forall v\in V$$

(2.38) $$T_i \ge T_j + st_j + t^v_{ji} - M_T s_{ij} - M(2-Y_{iv}-Y_{jv}) \qquad \forall i\in I,\forall p\in P,\forall v\in V$$

(2.39) $$TV_v \ge T_i + st_i + t^v_{ip} - M_T(2-X_{pv}-Y_{iv}) \qquad \forall i\in I,\forall p\in P,\forall v\in V$$

(2.40) $$\Delta a_i \le a_i - T_i \qquad \forall i\in I$$

(2.41) $$\Delta b_i \ge T_i - b_i \qquad \forall i\in I$$

(2.42) $$\Delta T_v \ge TV_v - tv_v^{\max} \qquad \forall v\in V$$

(2.43) $$\sum_{i\in I} w_i Y_{iv} \le q_v \sum_{p\in P} x_{pv} \qquad \forall v\in V$$

(2.44) $$X_{pv}\in\{1,0\};\ Y_{iv}\in\{1,0\};\ s_{ij}\in\{1,0\};\ c_i, T_i, \Delta a_i, \Delta b_i, \Delta T_v, {}_iCV_v, TV_v, \in \mathrm{R}$$

La restricción (2.30) asegura que cada cliente es atendido por sólo un vehículo, la restricción (2.31) que cada vehículo debe de ser asignado a lo más a un depósito, las restricciones (2.32)-(2.34)establecen que si el cliente j es atendido inmediatamente después del cliente i, entonces el costo de llegada para el cliente j debe de ser mayor a al costo de llegada al cliente i mas el costo de servicio para i mas el tiempo de ir de i a j, la restricción (2.35) establece que el tiempo total de la ruta debe de ser mayor al tiempo de llegada al último cliente en la ruta más el tiempo de servicio para éste cliente más el tiempo de ir del cliente i al depósito p y las restricciones (2.36)-(3.38) que si el cliente j es atendido inmediatamente después del cliente i, entonces el tiempo de llegada para el cliente j debe de ser mayor a al tiempo de llegada al cliente i mas el tiempo de servicio para i mas el tiempo de ir de i a j, la (2.39) son análogas a las restricciones (2.36)-(3.38) y establecen que el tiempo total de la ruta debe de ser mayor al tiempo de llegada al último cliente en la ruta más el tiempo de servicio para éste cliente más el tiempo de ir del cliente i al depósito p, las restricciones (2.40) y (2.41) verifican el cumplimiento de las ventanas de tiempo y en caso contrario se calcula el tiempo de espera, las restricciones (2.42) y (2.43) son para la duración máxima de cada ruta y la capacidad de carga de los vehículos respectivamente, y por último la restricción (2.44) establece la naturaleza de las variables.

La relevancia de presentar este modelo, radica en que en este trabajo se propone trabajar con una adaptación de éste modelo para el caso de MDVRPTW con ventanas de tiempo suaves, que se verá más adelante en el capítulo 3; mientras tanto la siguiente sección de este capítulo presenta algunas metaheurísticas con las que se han trabajado el MDVRP, VRPTW y MDVRPTW.

2.3 Metaheurísticas.

En esta sección se presenta un breve repaso de las heurísticas utilizadas para resolver tanto el VRPTW como el MDVRP y por supuesto el problema a tratar en este trabajo MDVRPTW.

La metaheurística Procedimientos de Búsqueda Miopes Aleatorizados y Adaptativos (GRASP, por sus siglas en inglés) propuesta por Resende, M. G., y Velarde, J. L. G. (2003), es una de las más famosas en la literatura, ya que parte de un procedimiento simple, un procedimiento miope aleatorizado en su primera fase de construcción y en la segunda fase, a la solución inicial se le aplica un algoritmo de búsqueda local. Debido a que este proceso se lleva a cabo un determinado número de veces, se le conoce como un método de multiarranque. La mejor solución de cada iteración se va guardando y la mejor de todas las iteraciones se devuelve como la mejor solución del metaheurístico.

El procedimiento miope que se mencionó, es un algoritmo de construcción en el que en cada iteración agrega a la solución en construcción, el elemento que más aporte en cierta función de ganancia definida por el usuario.

Según Marinakis, Y., y Athanasios M. (2007) indican que entre los trabajos publicados usando la metaheurística GRASP se encuentran los trabajos Chaovalitwongse, W., et. al. (2003), donde se utilizó la metaheurística GRASP al VRPTW y se proponen nuevos técnicas de mejora de soluciones; y el trabajo de Kontoravdis y Brand (1995) donde también se aplicó GRASP para el VRPTW.

Además de la metahurística GRASP, muchas otras heurísticas con las que se ha trabajado el VRP y sus variantes desde que fue introducido por Dantzig et. al. (1959) dando algunas de ellas excelentes resultados para los diferentes tipos problemas de ruteo, incluso ha habido heurísticas construidas para un tipo específico de problema que han resultado fácilmente adaptables para otras variantes del VRP como el caso de la heurística de Búsqueda Tabú

implementada por Cordeau et. al. (1997) y que fue mencionada en la sección anterior, para resolver en MDVRP junto con otra variante del VRP y el PTSP (Periodic Traveling Salesman Problem). Búsqueda Tabú es una metaheurística propuesta por Glover y Manuel Laguna, que utiliza estrategias "inteligentes", como procedimientos de aprendizaje y explota la historia del proceso de resolución del problema, tomando 4 dimensiones: propiedad de ser reciente, frecuencia, calidad e influencia; que además fue utilizada por Paraskevopoulos, D., et al. (2008), ya mencionado, junto con otra metaheurística, Búsqueda Reactiva de Entornos Variables (RVNS, por sus siglas en inglés) formando así una metaheurística hibrida que consiste de 2 etapas multiarranque. La primera etapa construye varias soluciones iniciales usando una heurística de construcción en semiparalelo y en la segunda etapa se utiliza el algoritmo de la metaheurística hibrida ReVNTS (Reactive Variable Neighborhood Search with Tadu Search) donde se aplica TS dentro del RVNS para mejorar las soluciones.

RVNS es una variante de la metaheurística de Búsqueda por Entornos Variables (VNS, por sus siglas en inglés), propuesta por Mladenović, N., y Hansen, P. (1997), la cual parte de una solución dada en una etapa anterior, para buscar mejorarla mediante cambios sistemáticos de entornos o vecindarios de la solución actual dentro de una búsqueda local.

Polacek, M., et. Al. (2004) abordó el MDVRPTW utilizando la metaheurística VNS después de una primera etapa para construir una solución inicial y posteriormente utilizó el VNS en tres pasos: agitación, búsqueda local y mover o no; aplicando así el VNS como en una etapa de mejora.

Kuo, Y., y Wang, C. C. (2012) utilizaron VNS para el Problema de Rutas de Vehículos con Múltiples Depósitos, Ventanas de Tiempo y Costos de Carga (MDVRPLC) y propusieron las siguientes heurísticas de búsqueda, para la búsqueda por entornos: inserción de nodo, intercambio de nodo, intercambio de arco e intercambio de sección.

A continuación se explican brevemente estas 3 heurísticas de búsqueda local.

Inserción de nodo: consiste en eliminar aleatoriamente un nodo de una ruta e insertarlo ese nodo dentro de otros dos clientes en la misma ruta o en cualquier otra.

Intercambio de arcos: se seleccionan aleatoriamente 2 arcos que conectan cada uno a un par de clientes (nodos) sucesivos de la misma ruta o de diferentes rutas y se intercambian.

Movimiento 2-opt: consiste en eliminar dos arcos y reconectar los dos caminos resultantes de una manera diferente para obtener una nueva ruta.

* Las *metaheurísticas* son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. Melián, B., et. al. (2003).

CAPÍTULO 3. MDVRPTW, modelos y heurísticas utilizadas**.**

En este capítulo se presentan los modelos desarrollados para tratar el MDVRPTW con las especificaciones propias de este trabajo, dadas en el capítulo anterior. Además, se explican los algoritmos desarrollados para la resolución del problema que da motivo a esta investigación.

3.1 Modelos.

Como ya se ha mencionado varias veces el problema a tratar en esta investigación es el problema de rutas de vehículos con múltiples depósitos y ventanas de tiempo, el cual consiste en encontrar un conjunto de rutas que partan de los diferentes m depósitos y den servicio a los n clientes buscando cumplir con las especificaciones de horarios $[e_i, l_i]$ dadas por los mismos, esto teniendo un número especifico S de vehículos para cada depósito y capacidad de carga y duración de ruta Q y r_k (respectivamente) limitados para los vehículos. En el caso en que no se cumpla con las ventanas de tiempo de los clientes, el planteamiento propuesto para abordar este problema permite que en tal caso se incurra en penalizaciones $P_i^1 y\, P_i^2$ por llegar el vehículo s en un tiempo t_i^s antes del inicio e_i de la ventana de tiempo del cliente i o después del cierre l_i respectivamente.

3.1.2 Modelo A

A continuación se presenta el modelo desarrollado en este trabajo para el MDVRPTW basado en la formulación hecha para el VRPTW, basado en un grafo $G = (V, E)$, donde V es el conjunto de nodos y E es el conjunto de arcos; en este trabajo se hace una subdivisión para el conjunto de nodos de manera que se divide en el subconjunto de L depósitos y R clientes, esto es, $V_L = \{1,2, \dots, L\}$ $V_R = \{L+1, L+2, \dots, L+R\}$, entonces $V = V_L \cup V_R$, para cada arco $(i, j) \in V$ se cuenta con las tiempos d_{ij} y los costos c_{ij} por ir de i a j;

además se tiene el conjunto de S vehículos homogéneos disponibles para cada depósito, donde cada vehículo cuenta con una capacidad Q, que por ser flota homogénea, es la mima para todos, al igual que el tiempo máximo de duración de la ruta de cada vehículo r_k (cada vehículo sólo hace una ruta). Para cada cliente i se tiene una demanda b_i y un tiempo de servicio f_i , donde $f_i = 0 \ \forall i \in V_L$; además de su ventana de tiempo $[e_i, l_i]$ y asociadas a la ventana de tiempo se tienen las multas P_i^1 y P_i^2 por incumplimiento de la ventana de tiempo de cada cliente.

Se utilizan las siguientes variables binarias: x_{ij}^{sl} igual a 1 si el vehículo s va del nodo i al j habiendo salido del depósito l y cero en otro caso, z_i^{s1} igual a 1 si el vehículo s llega el cliente i antes del inicio e_i de su ventana de tiempo y 0 en otro caso, de manera análoga también se define la variable z_i^{s2} igual a 1 si el vehículo s llega después del cierre de la ventana de tiempo l_i y cero en otro caso. También están las variables "auxiliares" continuas: w_{ij}^{sl} indica la carga del vehículo s al ir del nodo i al j habiendo salido ese vehículo del depósito l; el tiempo en que el vehículo s llega al cliente i es manejado por la variable continua t_i^s .

(3.1) $$\min_{x_{ij}^{sl},\, z_i^{s1}, z_i^{s2}, t_i} \quad \sum_{\substack{l\in V_L \\ s\in S}} \sum_{i,j\in V} c_{ij}^{11} x_{ij}^{sl} \;+\; \sum_{\substack{i\in V_R \\ s\in S}} P_i^1 z_i^{s1}\left(e_i - t_i\right) + \sum_{\substack{i\in V_R \\ s\in S}} P_i^2 z_i^{s2}\left(l_i - t_i\right)$$

s.a.

(3.2) $$\sum_{s\in S_l} \sum_{j\in V_R} x_{lj}^{sl} \le |S| \qquad \forall l \in V_L$$

(3.3) $$\sum_{r\in V_R} b_r \sum_{j\in V_R} x_{rj}^{sl} \le Q \qquad \forall s \in S\ \forall l \in V_L$$

(3.4) $$\sum_{j\in V_R} x_{lj}^{sl} = \sum_{j\in V_R} x_{jl}^{sl} \le 1 \qquad \forall s \in S\ \forall l \in V_L$$

(3.5) $$\sum_{\substack{i\in V_R \\ s\in S \\ l\in V_L}} x_{ir}^{sl} = \sum_{\substack{i\in V_R \\ s\in S \\ l\in V_L}} x_{ri}^{sl} = 1 \qquad \forall r \in V_R$$

(3.6) $$\sum_{j\in V_L} x_{jl}^{sl} = \sum_{j\in V_L} x_{lj}^{sl} = 0 \qquad \forall s \in S\ \forall l \in V_L$$

(3.7) $$0 \le w_{ij}^{sl} \le Q\, x_{ij}^{sl} \qquad \forall i, j \in V,\ \ \forall s \in S,\ \ \forall l \in V_L$$

(3.8) $$\sum_{\mathrm{j}\in \mathrm{V_R}} \sum_{s\in S} (x_{jr}^{sl} w_{jr}^{sl} - x_{rj}^{sl} w_{rj}^{sl}) = b_r \qquad \forall r \in V_R,\ \ \forall l \in V_L$$

(3.9) $$\sum_{\substack{i,j\in V_R \\ i\ne j}} x_{ij}^{sl} (d_{ij} + f_i + u_i^s) \le r_k \qquad \forall s \in S,\ \ \forall l \in V_L$$

(3.10) $$\sum_{\mathrm{s}\in \mathrm{S}} t_l^s = \sum_{\mathrm{s}\in \mathrm{S}} u_l^s = 0 \qquad \forall l \in V_L$$

(3.11) $$\sum_{\substack{i\in V_R \\ i\ne j}} x_{ij}^{sl} (t_i^s + d_{ij} + f_i + u_i^s) \le t_j^s \qquad \forall j \in V_R,\ \ \forall s \in S$$

(3.12*a*) $$e_i - \sum_{s\in S} t_i^s \le l_i \sum_{s\in S} z_i^{s1} \qquad \forall i \in V_R$$

(3.12*b*) $$\sum_{s\in S} t_i^s - e_i \le l_i (1 - \sum_{s\in S} z_i^{s1}) \qquad \forall i \in V_R$$

(3.13*a*) $$l_i - \sum_{s\in S} t_i^s \le r_k \sum_{s\in S} z_i^{s2} \qquad \forall i \in V_R$$

(3.13*b*) $$\sum_{s\in S} t_i^s - l_i \le r_k (1 - \sum_{s\in S} z_i^{s2}) \qquad \forall i \in V_R$$

(3.14) $$x_{ij}^{sl} \in \{0,1\}, \quad z_i^{s1} \in \{0,1\}, \quad z_i^{s2} \in \{0,1\}, \quad t_i^s \in R,\ w_{ij}^{sl} \in R$$

Cabe aclarar que las variables binarias z_i^{s1} y z_i^{s2} son parte de la propuesta de este trabajo, ya que en la literatura se encuentran pocos trabajos sobre el VRPTW con ventanas de tiempo suaves y los que se encontraron manejan las penalizaciones en la función objetivo con funciones que minimizan el máximo tiempo de espera en cada cliente; por lo que se utilizan dichas variables en la función objetivo de los dos modelos que se proponen en este trabajo, como indicadores para aplicar las multas en la función objetivo en el caso en que el vehículo incumpla con las ventanas de tiempo tomando un valor de uno si ese es el caso y cero en otro caso.

Observe que la función objetivo (3.1) no sólo minimiza el costo del trayecto total de todas las rutas si no también el costo por incumplimiento de las ventanas de tiempo de los clientes, este costo es proporcional al tiempo de incumplimiento, ya sea que el vehículo haya llegado antes o después del inicio o cierre de la ventana. La restricción (3.2) implica que el número de vehículos utilizados no debe exceder al número de vehículos disponibles para cada depósito, la restricción (3.3) es de la capacidad de los vehículos, la carga total de la ruta no debe exceder a las capacidad del vehículo, la restricción (3.4) asegura que cada vehículo sale y regresa al mismo depósito, la restricción (3.5) preserva el flujo, es decir si es vehículo s llega al cliente i también sale de él, la restricción (3.6) evita que un vehículo vaya de un depósito a otro, mientras que las restricciones (3.7) y (3.8) tratan de la carga del vehículo y las demandas de los clientes, la (3.7) indica que la carga del vehículo cuando pasa por el arco (i, j) no debe de exceder la capacidad del mismo y en la (3.8) que la carga que se deja en el cliente r debe ser igual a la demanda del cliente; la restricción (3.9) establece que la duración total de la ruta no debe exceder al máximo tiempo permitido para la misma, la restricción (3.10) marca que los tiempos de arribo (o salida) de los clientes deben de ser igual a cero, para la restricción (3.11) si el cliente j es atendido por el vehículo s después del cliente i entonces el tiempo de llegada al cliente j debe ser mayor que el del cliente i, las restricciones (3.12a)-(3.13b) verifican el cumplimiento de las ventanas de tiempo de los clientes , si estas no se cumplen, entonces se penalizan en la función objetivo, por último la restricción (3.14) es de la naturaleza de la variables.

3.1.3 Modelo B

El siguiente modelo es una adaptación del modelo de Dondo, R., y Cerdá, J. (2007) para el MDVRPTW con flota de vehículos heterogénea, como se comentó en el capítulo 2. Una de las adaptaciones que se hicieron es: en lugar de manejar dos tipos de variables similares C_i que es el costo distancia acumulada (recorrida) hasta llegar al cliente i y el tiempo T_i acumulado hasta llegar al cliente i, sólo se maneja en el modelo que se propone en este trabajo, la variable T_i ya que si se pueden considerar los costos de trayecto equivalentes o proporcionales a los tiempos acumulados, por lo que al suprimir la variable C_i también se elimina la variable CV_v que es el costo total de la distancia total recorrida (medida en unidades de tiempo) por el vehículo v y nos quedamos entonces con T_i y TV_v que es la duración total de la ruta hecha por el vehículo v, de esta manera resultan innecesarias las restricciones (2.32)-(2.34) del modelo de Dondo, R., y Cerdá, J. (2007), presentado en el capítulo anterior. Además se cambiaron las restricciones (2.40) y (2.41) concernientes a las ventanas de tiempo, éstas influyen en la función objetivo (2.29) de manera que en ella el costo de multas que se suma es el mismo para cuando se incumple la ventana de tiempo ya sea por arriba o por abajo, según el planteamiento de Dondo, R., y Cerdá, J. (2007), si $\Delta a = 0$ y $\Delta b = 0$ en las restricciones (2.32)-(2.34) entonces el modelo considera ventanas de tiempo estrictas y se incluyen tiempos de espera si es que el vehículo arriba antes del inicio de la ventana se tiempo, en el caso de ventanas de tiempo suaves Δa y Δb pueden ser mayor o igual que cero y el servicio se puede iniciar al tiempo en que el vehículo arribe, pero si incumple la ventana de tiempo entonces se penaliza; el cambio que se hizo fue para incluir en su lugar restricciones en las ventanas de tiempo suaves similares a las restricciones (3.12a)-(3.13b) presentadas en el modelo B las cuales utilizan las variables binarias z_i^1 y z_i^2. Por último se agregó una restricción para el número de vehículos disponibles para cada depósito.

Para este modelo B, los conjuntos y parámetros se definen igual que el de Dondo, R., y Cerdá, J. (2007), a excepción de las variables que como ya se comentó arriba, se omitieron. Las variables para este modelo son: $X_{\rho v}$ igual a 1 si el vehículo v es asignado al depósito ρ y cero en otro caso; Y_{iv} igual a 1 si el cliente i es asignado al vehículo v y cero en otro caso, y para llevar el seguimiento del orden en que son visitados los clientes se define (al igual que en el modelo de Dondo, R., y Cerdá, J. (2007)) la variable s_{ij} igual a 1 si el cliente i es visitado antes del cliente j y cero en otro caso; las variables z_i^1 igual a uno si se incumple la ventana de tiempo del cliente i por abajo, es decir, en su hora de inicio e_i y cero si no, y la variable z_i^2 igual a uno si se incumple la ventana de tiempo del cliente i por arriba, es decir, en su hora de cierre e_i y cero si no. También se definen las variables auxiliares continuas T_i que representa el tiempo acumulado hasta llegar al cliente i y la variable TV_v que es la duración total de la ruta hecha por el vehículo v.

También se incluyen al igual que en el modelo presentado por Dondo, R., y Cerdá, J. (2007), con tiempos t_{ij}^v que son representan el tiempo mínimo de ir del cliente i al cliente j mediante el vehículo v , de manera que el tiempo t_{pi}^v representa el mínimo tiempo de ir del depósito p al cliente i mediante el vehículo v y de la misma manera t_{ip}^v representa el mínimo tiempo de ir del cliente i al depósito p mediante el vehículo v , cuando éste termina su ruta.

A continuación se presenta el modelo B, una vez hechas estas aclaraciones que se consideraron pertinentes para la explicación del modelo.

(3.15) $$\min_{X_{pv},Y_{iv},TV_v} \quad \sum_{v\in V} TV_v + \sum_{i\in I} P_i^1 z_i^1 (e_i - T_i) + \sum_{i\in I} P_i^1 z_i^2 (T_i - l_i)$$

Sujeto a :

(3.16) $$\sum_v Y_{iv} = 1 \qquad \forall i \in I$$

(3.17) $$\sum_p X_{pv} \le 1 \qquad \forall v \in V$$

(3.18) $$\sum_v X_{pv} \le |S| \qquad \forall p \in P$$

(3.19) $$\sum_i Y_{iv} b_i \le Q \qquad \forall v \in V$$

(3.20) $$T_i \ge t_{pi}^v (X_{pv} + Y_{iv} - 1) \qquad \forall i \in I, \forall p \in P, \forall v \in V$$

(3.21a) $$T_j \ge T_i + st_i + t_{ij}^v - M_T (1 - s_{ij}) - M(2 - Y_{iv} - Y_{jv}) \qquad \forall i \in I, \forall p \in P, \forall v \in V$$

(3.21b) $$T_i \ge T_j + st_j + t_{ji}^v - M_T s_{ij} - M(2 - Y_{iv} - Y_{jv}) \qquad \forall i \in I, \forall p \in P, \forall v \in V$$

(3.22) $$TV_v \ge T_i + st_i + t_{ip}^v - M_T (2 - X_{pv} - Y_{iv}) \qquad \forall i \in I, \forall p \in P, \forall v \in V$$

(3.23) $$r_v \ge TV_v \qquad \forall v \in V$$

(3.24a) $$e_i - T_i \le r_v z_i^1 \qquad \forall i \in I$$

(3.24b) $$T_i - e_i \le r_v (1 - z_i^1) \qquad \forall i \in I$$

(3.25a) $$T_i - l_i \le r_v z_i^2 \qquad \forall i \in I$$

(3.25b) $$l_i - T_i \le r_v (1 - z_i^2) \qquad \forall i \in I$$

(3.26) $$X_{pv} \in \{1,0\}, \quad Y_{iv} \in \{1,0\}, \quad s_{ij} \in \{1,0\}, \quad z_i^1 \in \{1,0\}, \quad z_i^2 \in \{1,0\}, \quad T_i \in \mathrm{R}$$

La función objetivo (3.15) minimiza la suma del tiempo total de las rutas hechas por todos los depósitos junto con las penalizaciones por incumplimiento de las ventanas de tiempo proporcionales a las diferencias de tiempo entre que llegó el vehículo y los márgenes de las ventanas de tiempo de cada cliente. La restricción (3.16) asigna un cliente a exactamente un vehículo, y la (3.17) asigna a cada vehículo a lo más a un depósito, la restricción (3.18) verifica que el número de vehículos usados por cada depósito no exceda al número de vehículos disponibles para el mismo, así mismo la restricción (3.19) verifica que la carga total del vehículo no exceda su capacidad, la restricción (3.20) establece que el servicio al

cliente debe ser mayor o igual al tiempo de llegada del vehículo v a dicho cliente, las restricciones (3.21a) y (3.21b) están relacionadas de manera que si el cliente i es visitado antes del cliente j entonces el tiempo en que se inicia el servicio (o el vehículo llega) al cliente i debe ser mayor al del cliente j, si no es así estas restricciones se vuelven redundantes, la restricción (3.22) asegura que el tiempo total de la ruta hecha por el vehículo sea mayor o igual al tiempo en que llega de regreso al depósito, la restricción (3.23) verifica que la duración total de la ruta no exceda a la duración máxima permitida para la ruta, las restricciones (3.24a)-(3.25b) al igual que en el modelo A verifican el cumplimiento de las ventanas de tiempo de los clientes , si estas no se cumplen, entonces se penalizan en la función objetivo, y por último la restricción (3.26) es de la naturaleza de la variables.

Aunque en ambos modelos se presentan los mismos tipos de restricciones, las variables que se plantean para ambos modelos difieren en el número de índices que maneja, por lo que las asignaciones en el modelo B de cliente a vehículo y vehículo a depósito difieren del modelo A que utiliza 4 índices para manejar la asignación.

3.2 Heurísticas

En esta sección se explica la estrategia de solución desarrollada en este trabajo para el MDVRPTW, la cual consiste en un algoritmo de dos etapas, la etapa I es la etapa de construcción de una solución inicial, para entonces introducir la etapa II que es la etapa de mejora.

Para la etapa I, se construyeron dos algoritmos de construcción de una solución inicial. Sin embargo, ambos algoritmos inician haciendo una asignación de clientes a depósitos de la misma manera, con una función de probabilidad:

$$3.2.1) \qquad P(i,j^*) = \frac{d(i,j^*)}{\sum_{i \in I} d(i,j^*)} \qquad \forall j \in J$$

Donde I es el conjunto de clientes y J es el conjunto de depósitos.

Esta función da la probabilidad de que al depósito j se le asigne el cliente i, dividiendo la distancia del cliente i al depósito j entre la suma de las distancias de los demás clientes al depósito en cuestión.

Posteriormente para asignar cada cliente i a alguno de los L depósitos, se construyen L cotas para cada cliente, estas se construyen siendo la primera cota del cliente i, $C_{i0} = 0$ la siguiente cota es $C_{i1} = C_{io} + P(i,1)$, de esta manera se obtiene el rango de probabilidad asignación del cliente i para el primer depósito $[C_{io}, C_{i1}]$; entonces el rango de asignación del cliente i para el deposito j es $[C_{i,j-1}, C_{i,j}]$, donde $C_{i,j} = C_{i,j-1} + P(i,j)$. Una vez construidas las cotas por cliente con base en las probabilidades dadas por la función 3.2.1), se utiliza un número aleatorio na_i para cada cliente y se busca dentro de cuál de los L rangos se encuentra este número aleatorio, para asignar el cliente i al depósito j correspondiente a dicho rango.

Aplicando este procedimiento a todos los clientes, se tienen repartidos. Entonces, los R clientes entre los L depósitos, con base en la función de probabilidad 3.2.1).

Para continuar con la etapa de construcción, se crearon 2 algoritmos distintos, el primero basado en la idea de rutear primero y agrupar después y el segundo en una heurística greedy aleatoria para la creación de las rutas.

3.2.1 Construcción inicial I

En la construcción inicial I, lo que se hace es construir una mega ruta para cada depósito con todos los clientes asignados a cada uno de ellos tomando de referencia los promedios de las ventanas de tiempo, para usar el cliente con mayor promedio de ventana de tiempo como la semilla para construir la mega ruta y posteriormente ir insertando los clientes según su ventana de tiempo entre el depósito y el cliente semilla; una vez que tenemos la mega ruta, se procede a hacer cortes* en la misma (buscando factibilidad), para obtener las rutas pertenecientes a ése depósito.

Asignación.

Matriz de probabilidades.

Mega ruta.

i) Para cada cliente no ruteado y asignado a ese depósito, buscar la posición de inserción con mayor ahorro para ese cliente y si es FACTIBLE, si lo es entonces, calcular AHORROS.

ii) Para cada cliente encontrar la mejor posición para insertarlo.

iii) Buscar el cliente con mejor valor de ahorro e insertarlo en la ruta.

Cortes.

Factibilidad, según:

- Duración máxima permitida para las rutas.
- Capacidad.

Para la parte de asignación la idea de la función de probabilidad se obtuvo a partir del trabajo de Kuo, Y. (2013) donde utiliza una función de probabilidad distinta para hacer las asignaciones; por otra parte la construcción de la mega ruta la inicialización usando un cliente semilla se basó en la explicación dada por Bräysy, O., y Gendreau, M. (2005) acerca del trabajo de Solomon, M. M. (1987) y el ordenamiento de los clientes según el promedio de sus ventanas de tiempo para seleccionar el cliente semilla fue tomado el trabajo de Polacek, M., et. al. (2004).

En Solomon, M. M. (1987) se explican 3 posibles enfoques de aplicar la heurística de inserción allí presentada, en este trabajo se utilizó la tercera opción, donde para la actual solución en construcción $(i_o, i_1, i_2, \dots, i_m)$ se calcula el mejor lugar de inserción en la ruta actual para todos los cliente que aún no se han insertado en la ruta, para después obtener el cliente con el mejor costo de inserción; esto se hace considerando la siguiente ecuación:

$$3.2.2) \quad c(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) + \alpha_3 c_{13}(i,u,j)$$

Donde $\alpha_1 + \alpha_2 + \alpha_3 = 1, \ \alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0$

Y $c_{11}(i,u,j) = d(i,u) + d(u,j) - \mu d(i,j), \quad \mu \geq 0$

$c_{12}(i,u,j) = b_{j_u} - b_j$, donde b_{j_u} es el nuevo tiempo en que se inicia el servicio al

Cliente j dado que se insertó el cliente u.

$c_{13}(i,u,j) = l_u - b_u$.

En la ecuación 3.2.2) se busca el costo de insertar el cliente u entre los clientes ya ruteados i y j, para esto se hace un balance utilizando los parámetros α_1, α_2 y α_3 entre las ecuaciones $c_{11}(i,u,j)$, $c_{12}(i,u,j) = b_{j_u}$ y $c_{13}(i,u,j)$; la ecuación $c_{11}(i,u,j)$ calcula el

posible ahorro generado en distancia si se inserta el cliente u entre los clientes i y j ya ruteados, la ecuación $c_{12}(i,u,j)$ calcula la diferencia de tiempo de iniciar el servicio para el cliente j dado que se insertó el cliente u, la ecuación $c_{13}(i,u,j)$ calcula la diferencia entre el tiempo en que se inicia el servicio al cliente u y el cierre de su ventana de tiempo.

Usando la ecuación 3.2.2) se busca para cada cliente no ruteado la mejor posición donde puede ser insertado, después se obtiene el cliente con mejor posición de inserción y se inserta a en la ruta actual.

Después de que se obtiene la mega ruta de la manera ya descrita, el siguiente paso es hacer los cortes a la mega ruta, utilizando 2 criterios de factibilidad: capacidad de los vehículos y distancia máxima permitida para las rutas, esto es cuando en la mega ruta se incumple con una de estas dos condiciones, se hace un corte y se inicia una nueva ruta con lo el resto de la mega ruta y así sucesivamente, esto se hace para las mega rutas de todos los depósitos y de esta manera se obtiene el conjunto de rutas factibles para los depósitos siendo éste conjunto la solución inicial con la que se trabajará en la etapa de mejora.

3.2.2 Construcción inicial II.

Para la segunda construcción inicial, a partir de que los clientes ya están asignados a los depósitos, el procedimiento a seguir es que por cada depósito crear S rutas, ya que sólo se dispone de S vehículos y cada vehículo sólo puede hacer una ruta. Entonces por cada depósito se construyen S rutas siguiendo un procedimiento greedy aleatorio que en base a una lista restringida de candidatos (RCL, pos sus siglas en inglés) con los mejores clientes candidatos no ruteados aún para cada ruta se escoge al azar uno de ellos para cada una de las rutas. Cabe aclarar que todos los clientes no ruteados se consideran para cada una de las S listas restringidas de candidatos del depósito en turno y cuando un cliente es seleccionado de una de las RCL e incluido en una de las S rutas, entonces deja de ser

candidato para esa y todas las demás listas en los siguientes pasos. El procedimiento para cada depósito concluye cuando todos los clientes han sido asignados.

El procedimiento se puede resumir de la siguiente manera:

1.- Se selecciona aleatoriamente un depósito para iniciar las S rutas.

2.- Se toman S listas restringidas de candidatos (RCL).

3.- Se inserta al azar un cliente perteneciente a cada RCL a cada ruta.

4.- Si ya no hay más clientes por insertar, entonces el procedimiento termina.

Para la construcción de las RCL, se puede proceder de dos maneras, una de ellas es seleccionar a los clientes que integraran la RCL utilizando como criterio un tamaño específico nc para la lista y de entre los clientes no ruteados seleccionar los mejores nc candidatos para la ruta en cuestión.

O bien, otra manera de seleccionar los mejores clientes que integren la RCL para una de las rutas en construcción es seleccionando los clientes no ruteados aun que contribuyen a la ruta actual en un porcentaje no menor a α% del mejor candidato, mediante la función:

$$3.2.3) \quad f(i) = c(i_*) + \alpha\big(c(i^*) - c(i_*)\big)$$

Donde $c(i) = t(i) + P_1 z_1(e_i - t_i) + P_2 z_2(t_i - l_i)$ esto es, el costo de insertar el cliente i a la ruta es igual al tiempo de llegada al cliente i si se inserta más la multa P_1 por el tiempo de incumplimiento de la ventana de tiempo por abajo si es que si se incumplió la ventana de tiempo por abajo (en tal caso $z_1 = 1$ y cero si no), más la multa P_2 por el tiempo de

incumplimiento de la ventana de tiempo por arriba si es que si se incumplió la ventana de tiempo por arriba (en tal caso $z_2 = 1$ y cero si no); además i_* es el cliente candidato con el menor costo de inserción y el cliente i^* es el cliente candidato con el mayor costo de inserción.

En este trabajo se utilizaron ambas opciones para la construcción de las RCL, las diferencias entre ellas se reflejan en los resultados encontrados en las experimentaciones hechas con esta construcción inicial II; tales resultados se muestran en el capítulo 4.

3.3 Mejora.

A continuación se explican brevemente estas 4 heurísticas de búsqueda local.

Reinserción de nodo: consiste en eliminar aleatoriamente un nodo de una ruta e insertarlo ese nodo entre otros dos clientes en la misma ruta o en cualquier otra.

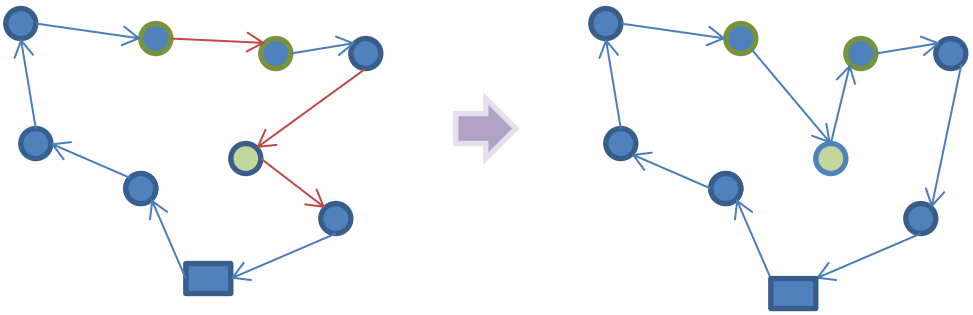

Fig. 1 Reinserción de nodo.

Intercambio de arcos: se seleccionan aleatoriamente 2 arcos que conectan cada uno a un par de clientes (nodos) sucesivos de la misma ruta o de diferentes rutas y se intercambian.

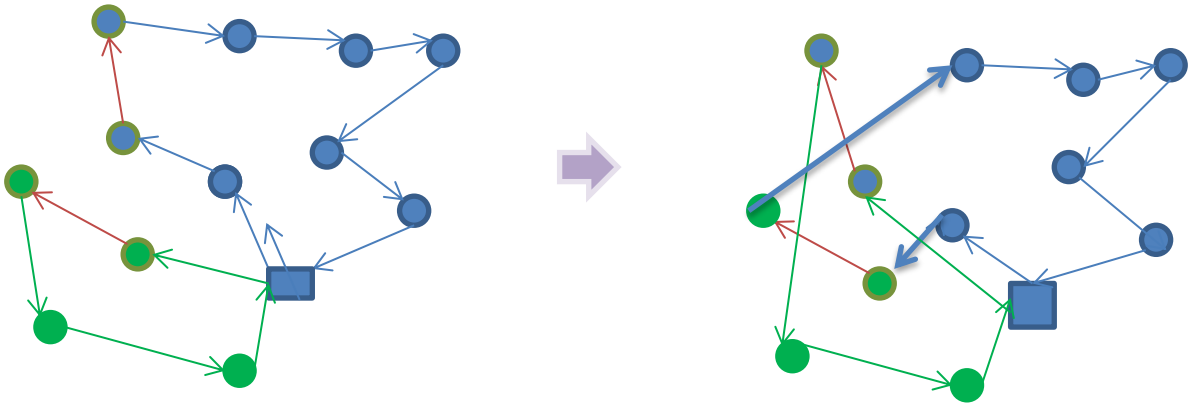

Fig. 2 Intercambio de arcos.

Movimiento 2-Opt: consiste en eliminar dos arcos y reconectar los dos caminos resultantes de una manera diferente para obtener una nueva ruta.

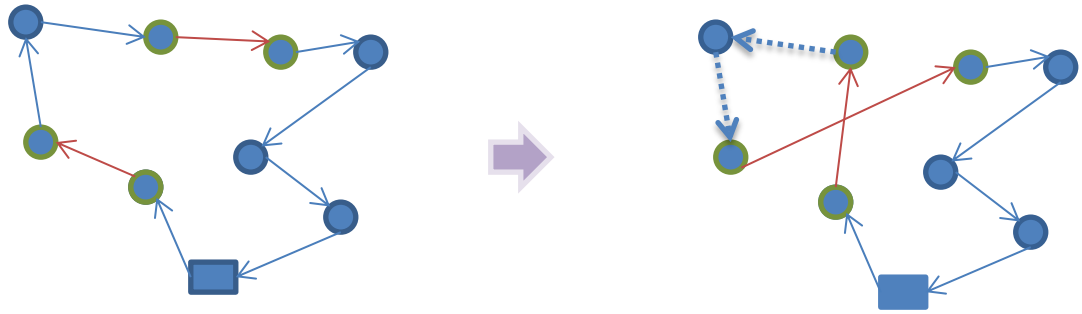

Fig. 3 Movimiento 2-Opt.

El movimiento de reinserción de nodos se aplicó tanto dentro de una misma ruta como entre rutas de un mismo depósito y entre rutas de dos depósitos. Sin embargo, para los movimientos de intercambio entre arcos y 2-Opt, los movimientos fueron aplicados solo entre rutas de un mismo depósito y entre rutas entre depósitos.

CAPÍTULO 4. EXPERIMENTACIÓN.

En la etapa de experimentación se hicieron pruebas para ambos algoritmos que fueron implementados en C++ y la experimentación desarrollada en una computadora con procesador Intel ® Core™ i5 1.80GHz, 6GB RAM.

Utilizando las instancias para el MDVRPTW disponibles en http://neumann.hec.ca/chairedistributique/data/, las cuales consisten en conjunto de datos que van desde los 48 clientes y 4 depósitos, hasta 288 clientes y 6 depósitos, siendo en total 20 instancias que consideran una flota de vehículos homogénea con una capacidad de carga limitada Q y duración máxima T para las rutas; y además difieren en los tamaños de las ventanas de tiempo de los clientes, por ejemplo, en las instancias de la 11 a la 20 la amplitud de las ventanas de tiempo es más grande que en las primeras 10 instancias por lo que en estas instancias se cuenta con menos vehículos que en las otras instancias de igual tamaño pero con la amplitud de las ventanas de tiempo más chica.

En la siguiente tabla se presentan los resultados de las pruebas que se hicieron para las 20 instancias utilizando solamente la etapa constructiva, es decir los dos algoritmos sin implementar los movimientos de mejora, mostrando en las últimas dos columnas el valor de la mejor solución inicial encontrada por ambos algoritmos de un total de 100 pruebas, junto con el promedio de los valores encontrados para las soluciones iníciales dadas en estas 100 pruebas.

INSTANCIA	N	m	T	ALGORITMO I	ALGORITMO II
01	48	4	2	1151.01	3994.72
02	96	4	3	1690.54	1473.45
03	144	4	4	2066.50	2641.99
04	192	4	5	1273.01	2041.43
05	240	4	6	1959.61	2675.23
06	288	4	7	1389.02	1081.98
07	72	6	2	862.65	1522.54
08	144	6	3	1344.70	1114.21
09	216	6	4	1819.42	3511.71
10	288	6	5	989.34	2728.91
11	48	4	1	1893.32	2636.48
12	96	4	2	2636.48	4393.65
13	144	4	3	2069.98	2709.26
14	192	4	4	1269.17	1639.62
15	240	4	5	1513.81	3051.75
16	288	4	6	1647.68	3174.21
17	72	6	1	844.47	1582.64
18	144	6	2	1962.42	2736.16
19	216	6	3	2056.27	2994.14
20	288	6	4	1958.39	2866.65

Para la experimentación con la segunda etapa, la etapa de mejora, los resultados se comparan con los presentados por Polacek, M., et. al. (2004) para el MDVRPTW con flota homogénea de vehículos, donde se implementó una metaheurística de búsqueda por entornos variables (VNS, por sus siglas en inglés) utilizando las mismas instancias para su experimentación.

INSTANCIA	N	m	T	ALGORITMO I	ALGORITMO II	VNS
01	48	4	2	945.43	5088.78	1083.98
02	96	4	3	954.09	3377.77	1762.21
03	144	4	4	1600.07	1226.10	2374.36
04	192	4	5	2164.26	4137.95	2858.20
05	240	4	6	2058.95	3706.63	3040.84
06	288	4	7	2120.02	2027.09	3758.36
07	72	6	2	975.29	4247.46	1522.65
08	144	6	3	1199.82	2746.05	2103.89
09	216	6	4	1983.86	2713.66	2783.95
10	288	6	5	1861.10	3073.97	3577.28
11	48	4	1	882.74	1501.93	1005.73
12	96	4	2	835.91	1906.77	1487.64
13	144	4	3	1154.37	4190.56	2014.02
14	192	4	4	1264.07	2557.18	2221.17
15	240	4	5	1415.97	2867.68	2494.82
16	288	4	6	2188.14	2405.95	2939.20
17	72	6	1	1439.92	944.88	1239.13
18	144	6	2	1332.09	2179.63	1796.21
19	216	6	3	1924.42	3816.21	3079.73
20	288	6	4	2194.05	2808.58	2268.10

CAPÍTULO 5. CONCLUSIONES.

5.1 Conclusiones.

El Problema de Rutas de Vehículos es un problema extensamente estudiado en el área de optimización, del cual han surgido gran número de variantes, debido a particularidades en las restricciones del problema o incluso de la función objetivo.

La variante con la que se trabajó en esta tesis, es el Problema de Rutas de Vehículos con Múltiples Depósitos y Ventanas de Tiempo que en este trabajo se consideraron como suaves. Para este problema se hizo una revisión de literatura, abarcando desde los primeros trabajos hechos para el VRP hasta los hechos para el MDVRPTW. Cabe aclarar que en nuestra revisión de literatura se encontraron pocos trabajos hechos para el MDVRPTW con ventanas de tiempo suaves, siendo el caso de estrictas y para un solo vehículo el tipo de trabajos que más se encontraron en lo que respecta a nuestra revisión de literatura.

En esta tesis se propuso un modelo matemático para el problema en cuestión y que fue probado para instancias hasta de 30 nodos, teniendo un buen desempeño, dando soluciones óptimas.

El algoritmo computacional desarrollado en este trabajo de tesis consiste en dos etapas; la primera etapa es de construcción de una solución inicial, y la segunda etapa es de mejora.

Para la primera etapa se siguió un procedimiento de asignación de todos los clientes a los depósitos, para posteriormente crear una mega ruta para cada depósito con sus respectivos clientes asignados y después realizar cortes en las mega rutas considerando las restricciones de capacidad de carga de los vehículos y el tiempo de trabajo permitido para los vehículos por cada ruta, obteniendo así una solución inicial.

En la segunda etapa, se implementaron los movimientos de mejora de reinserción de nodos, intercambio de arcos y el movimiento 2-Opt. Los tres movimientos mencionados

fueron implementados entre rutas tanto del mismo depósito como entre depósitos; sólo el movimiento de reinserción de nodos fue implementado también para hacer cambios dentro de una misma ruta.

Éste algoritmo se probó con las mismas instancias con que se probó el modelo matemático, y se obtuvieron muy buenos resultados, ya que en ambos casos se llegó al óptimo, siendo el algoritmo computacional el que tardó menos en resolver el problema.

La segunda experimentación consistió en utilizar las instancias descritas en el capítulo 4 con tan sólo la primera etapa del algoritmo arrojando resultados en cuanto a tiempos. Sin embargo para la experimentación con las dos etapas fue necesario modificar el algoritmo para que trabajara con ventanas de tiempo estrictas. Al hacer las comparaciones con el trabajo previo se encontró que los resultados eran muy similares y en algunos casos mejores a los obtenidos en el trabajo previo con un algoritmo VNS.

5.2. Trabajo futuro.

Implementar un algoritmo GRASP para resolver el MDVRPTW con los mismos supuestos de este trabajo para obtener resultados con los cuales comparar el algoritmo con ventanas de tiempo suaves. Con el trabajo ya hecho, buscar resolver el MDVRPTW con flota heterogénea. De la misma manera, buscar resolver el MDVRPTW con más de una ruta por vehículo.

Referencias

[1] Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation science*,*39*(1), 104-118.

[2] Chaovalitwongse, W., Kim, D., & Pardalos, P. M. (2003). GRASP with a new local search scheme for vehicle routing problems with time windows. Journal of Combinatorial Optimization, 7(2), 179-207.

[3] Cordeau, J. et. al. (1997). A tabu search heuristic for Periodic and Multi-Depot Vehicle Routing Problems. (John Wiley & Sons, Inc. Networks).

[4] Crevier, B., Cordeau, J. F., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*,*176*(2), 756-773.

[5] Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem.*Management science*, *6*(1), 80-91.

[6] Dethloff, J. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR-Spektrum*, *23*(1), 79-96.

[7] Dondo, R., & Cerdá, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows.*European Journal of Operational Research*, *176*(3), 1478-1507.

[8] Fung, R. Y., Tang, J., & Zhang, J. (2009, July). A multi-depot vehicle routing problem with weight-related costs. In *Computers & Industrial Engineering, 2009. CIE 2009. International Conference on* (pp. 1028-1033). IEEE.

[9] Kallehauge, B., Larsen, J., Madsen, O. B., & Solomon, M. M. (2005). *Vehicle routing problem with time windows* (pp. 67-98). Springer US.

[10] Kontoravdis, G., and Bard, J.F., (1995), "A GRASP for the Vehicle Routing Problem with Time Windows", ORSA Journal on Computing, Vol. 7, pp. 10–23.

[11] Kuo, Y. (2013). Optimizing truck sequencing and truck dock assignment in a cross docking system. *Expert Systems with Applications*, *40*(14), 5532-5541.

[12] Kuo, Y., & Wang, C. C. (2012). A variable neighborhood search for the multi-depot vehicle routing problem with loading cost. *Expert Systems with Applications*, *39*(8), 6949-6954.

[13] LaLonde, Bernard J. "Supply Chain Evolution by the Numbers," Supply Chain Management Review, Vol. 2, No. 1, pp. 7-8, 1998.

[14] Marinakis, Y., & Migdalas, A. (2007). Annotated bibliography in vehicle routing.*Operational Research*, *7*(1), 27-46.

[15] Melián, B., Pérez, J. A. M., & Vega, J. M. M. (2003). Metaheurısticas: una visión global. Revista Iberoamericana de Inteligencia Artificial, 19, 7-28.

[16] Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, *23*(5), 377-386.

[17] Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097-1100.

[18] Paraskevopoulos, D. C., Repoussis, P. P., Tarantilis, C. D., Ioannou, G., & Prastacos, G. P. (2008). A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics*, *14*(5), 425-455.

[19] Polacek, M., Hartl, R. F., Doerner, K., & Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics*, *10*(6), 613-627.

[20] Potvin, J. Y., & Rousseau, J. M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, *66*(3), 331-340.

[21] Renaud, J., Laporte, G., & Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, *23*(3), 229-235.

[22] Resende, M. G., & Velarde, J. L. G. (2003). GRASP: Procedimientos de búsquedas miopes aleatorizados y adaptativos. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, *7*(19), 61-76.

[23] Solomon, M. (1987). Algorithms for the Vehicle Routing Problem with Time Window Constraints. Institute for Operations Research and the Management Sciences.

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

[25] Surekha, P., & Sumathi, S. (2011). Solution to multi-depot vehicle routing problem using genetic algorithms. *World Applied Programming*, *1*(3), 118-131.

[26] Toth, P., & Vigo, D. (Eds.). (2001). *The vehicle routing problem*. Siam.

CÓDIGO

```
//***LIBRERIAS***//

#include <stdio.h>

#include <math.h>

#include <conio.h>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#define random(num)(rand()%(num))



//*****FUNCIONES//////

using namespace std;

void fdistancias();

void fprobabilidades();

void fasignacion();

void fcadena();

void fbalance();

void fmegaruta();

void fcortes();

void fmejora();
```

```
void fdatos();


//*****VARIABLES***//

int
N,m,n,v,rk,i,j,h,k,l,M,na[100],d[100],st[100],inicio,horai[100],horaf[100],ind,cont,conta,u,longcad[4]
,cad[52],vr[100], r[100],demacum_1[100],demacum_2[100],asign[4][48],
dem[52],maclient[4],posmaclient[52],caa,mr[150],insert=1,pos;

double x[],y[], a, bc,vabs,p,sum,sum1,sum2,max1, mdc[52][52],
mdist[4][48],promc[48],dif[4][48],max[4][48],mprob[4][48],cota[5],ma[4],ma1,prom[52],b[150],aho
rro[200][200],baux[200][200],holgaux[200][200],c1[200][200],p1=-1,p2=-1,p3=-
1,costo_insert[155][155];;

float balance[52],tllegada_1[100],tsalida_1[100],tllegada_2[100],tsalida_2[100],distacum_1[100];



//****INICIO****//

int main()

{

system("Pause");

//********** LEER DATOS ----------//

fdatos();

//********** DISTANCIAS ----------//

fdistancias();

system("Pause");

//********** PROBABILIDADES ----------//
```

```
fprobabilidades();

system("Pause");

//********** ASIGNACION ----------//

fasignacion();

system("Pause");

//********** CADENA ----------//

fcadena();

//********** BALANCE ----------//

fbalance();

system("Pause");

//********** EMPIEZA MEGA-RUTA**********

fmegaruta();

system("Pause");

//********** CORTES ----------//

fcortes();

//********** MEJORA ----------//

fmejora();

system("Pause");

return 0;



}
```

```
void fdistancias()

{

    //(1) Matriz de distancias de partiendo de las posiciones en x,y*/

//printf("Distancias\n");




  for(i=0;i<N;i++)   //*******hacer for(i=0;i<N,i++) y los ultimos son depositos Y for(j=0;j<n,j++) osea a la dercha sólo clientes**********

    {



        for(j=0;j<N;j++)

        {

            a=x[i]-x[j];

            bc=y[i]-y[j];



            p1=a*a;

            p2=bc*bc;

            p=p1+p2;

            mdc[i][j]= sqrt(p);

               printf("%lf\t",mdc[i][j]);
```

```
        }


          // printf("\n\n\n ");
     }


   //printf("\n\n\n ");
   //system("Pause");
          // printf("\n\n\n");


}


void fprobabilidades()
{
   for(j=0;j<n;j++)//*****************  for(i=n;i<N;i++)  osea  los  depositos  están  al final**************


    {
       sum1=0;
       for(i=n;i<N;i++)
       {
          sum1=sum1+mdc[i][j];
          //printf("%lf\t",mdc[i][j]);
```

```
        }
         //printf("%lf\t",sum1);

        for(i=n;i<N;i++)
        {
          // printf("%lf\t",mdc[i][j]);
            mprob[i-n][j]=mdc[i][j]/sum1;  //*******mprob[i-n][j] *********
         //  printf("%lf\t",mprob[i-n][j]);
        }//printf("FIN Probabilidades\n");
       //  printf("\n");
       //  printf("\n");
     }



}

void fasignacion()
{

int piso_cota;
// printf("Asignación\n");
```

```
//printf("\n\naleatorios\n");
 /* for(j=0;j<=n;j++)
 {

   na[j]=rand()%50;
   printf("%d\t",na[j]);
 }
  srand(time(NULL));*/

//   printf("\n\n\n");
//   printf("\n\n");
//   printf("COTAS\n");
  for(j=0;j<n;j++)
  {
     cota[0]=0;
     // printf("%lf\t",cota[0]);
     for(i=0;i<m;i++)
     {
        cota[i+1]=(cota[i]+(mprob[i][j]*100));
       //  printf("%lf\t",cota[i+1]);
     }
```

```
    //  printf("fin de cotas\t\t");


//    printf("\n");



 //  printf("cotadep4 \t%lf\t\t",cota[m]);
   piso_cota=floor(cota[m]);
   na[j]=rand()%piso_cota;
   //na[j]=rand()%50;
//   printf("\t\taleatorio\t");
//   printf("%d\t",na[j]);
//   printf("\n\n\n\n");



     for(i=0;i<m;i++)
     {
        if(na[j]>=cota[i] && na[j]<cota[i+1])
          {
            asign[i][j]=1;
          }
        else
          {
```

```
            asign[i][j]=0;
           }
      }

 }


system("Pause");
       printf("\n\n\n\n\n\n\n");
    printf("asignaciín 1 y 0");
    for(i=0;i<m;i++)
     {
         printf("\n\n");
         for(j=0;j<n;j++)
         {
         printf("%d\t",asign[i][j]);

         }

     }
}
```

```
void fcadena()
{
    for(i=0;i<m;i++)//cuenta el num de clientes asignados a c/depot
    {
        cont=0;
        for(j=0;j<n;j++)
        {
          if(asign[i][j]==1)
          {
           cont=cont+1;
          }

        }
       longcad[i]=cont;

    }
    int may, dep=n;// Encuentra cual es el mayor numero de clientes asignados por depot
    may=longcad[0];
        for(i=1;i<m;i++)
        {
            if(may<longcad[i])
```

```
            {   dep=n+i;
                may=longcad[i];

            }
        }

        printf("\n\nlongcad\n");
for(i=0;i<m;i++)
        {
        printf("%d-",longcad[i]);

        }
//printf("\n\n");

   printf("\n\n");
printf("el depot con mas asignados es \t %d\t,%d",may,dep);

system("Pause");
int k=0;/**********************/

   for(i=n;i<N;i++)//rellena el vector cadena con los clientes asignados por depot
    {
```

```
        cad[k]=i;
        k=k+1;
        for(j=0;j<n;j++)
        {
          if(asign[i-n][j]==1)
          {

            cad[k]=j;
            k=k+1;
          }
        }cad[k]=i;
         k=k+1;
    }

 printf("\ncadena\n");

for(i=0;i<N+m;i++)
    {
        printf("%d\t",cad[i]);

    }   printf("fin de cadena\n");
```

```
for(i=0;i<N;i++)//Saca promedios de las TW para todos
 {
     sum=horai[i] + horaf[i] ;
     p=(sum/2);
     prom[i]=p;
     printf("//%f\t",prom[i]);
 }
        printf("\n\n promedios de las tw de los clientes\n");
        for(i=n;i<N;i++)
        { printf("\n");
         for(j=0;j<N;j++)
         {
           printf("%f\t",mdc[i][j]);
         //printf("//%f\t\t",prom[i]);

         }printf("\n\n");
         }
}

void fbalance()
{
```

```
 inicio=0;
 printf("balance");
 for(i=0;i<m;i++)
 {  for(j=inicio+0;j<longcad[i]+1;j++)
       {
        balance[j]=prom[j]+mdc[i][j];
        printf("%d-%f\t",j,balance[j]);
       }
       inicio=inicio+longcad[i]+1;
        printf("\n");
 }
         system("Pause");
         printf("\n\n");



inicio=0;


for(i=0;i<m;i++)
{
   caa=cad[inicio+1];
   ma[i]=prom[caa];
   for(u=inicio+1;u<=inicio+longcad[i];u++)//   se supone q ya lo corregì, pte checar
```

```
   {
     if(cad[u]<n)
     {
       if(ma[i]<=prom[cad[u]])
       {
         ma[i]=prom[cad[u]];
         maclient[i]=cad[u];
         posmaclient[i]=u;
       }
     }
   }printf("( %.2f )",ma[i]);
 inicio=inicio+longcad[i]+2;
}

printf("\n\n\n\n");
 printf("cliente con mayor prom para c/deposito\n");
for(i=0;i<m;i++)
        {
        printf("%.2f-",ma[i]);

        printf("%d\t",maclient[i])
        }
```

```
printf("mayor para c/deposito FIN\n");

printf("\n\n\n");

}


void fmegaruta()

{

  for(i=0;i<N;i++)

  {

     b[i]=0;

  }

  printf("\n\n\n");


  int
st[]={2,7,21,24,1,7,6,5,7,1,4,10,2,9,23,18,3,15,13,10,4,23,20,10,4,2,23,6,8,10,7,1,21,6,4,13,9,22,22,
18,10,25,23,4,9,17,10,17,0,0,0,0};


      for(i=0;i<N;i++)

    {

     printf("%d/",st[i]);

    } printf("\n\n\n");


  j=0;
```

```
 int w1=0,w2=0,w3=0, w4=0, w5=0;
  for(i=0;i<3*m;i=i+3)
  {
     mr[i]=n+j;
     w1=mr[i];
     b[i]=0;

     mr[i+1]=maclient[j];
     cad[posmaclient[j]]=-1;
     w2=mr[i+1];
     b[i+1]=st[w1]+mdc[n+j][w2]+b[i];
     mr[i+2]=n+j;
     w3=mr[i+2];
     b[i+2]=st[w2]+mdc[w2][n+j]+b[i+1];
     j++;

  }
   printf("\n\n");
inicio=0;

for(i=0;i<m;i++)
{
```

```
    insert=1;
 do{


   for(u=inicio+1;u<=inicio+longcad[i]+1;u++)// u<inicio+1+longcad[cad[i]] es dsd donde me quedé y hasta donde le sigo para ese  deposito
   {
      if(cad[u]<n && cad[u]!=-1)//si el que voy a verificar NO es deposito entonces le sigo
      {
      for(pos=inicio+1;pos<=inicio+insert+1;pos++)
       {
         if(horai[mr[pos-1]]<horai[cad[u]]&&horai[cad[u]]<horaf[mr[pos]])


          {
            costo_insert[cad[u]][pos]=mdc[mr[pos-1]][cad[u]]+mdc[cad[u]][mr[pos]];
            w4=pos;
            w5=pos-1;
            baux[cad[u]][pos]=b[pos-1]+st[mr[pos-1]]+mdc[cad[u]][mr[pos-1]];
            baux[mr[pos]][pos+1]=baux[cad[u]][pos]+st[cad[u]]+mdc[cad[u]][mr[pos]];


            holgaux[cad[u]][pos]=horaf[cad[u]]-baux[cad[u]][pos];

c1[cad[u]][pos]=p1*costo_insert[cad[u]][pos]+p2*baux[mr[pos]][pos+1]+p3*holgaux[cad[u]][mr[pos]];
```

```
        }
       }
      }
    }

int maycliente, mejpos[200],mpos[200],bestu,ma1;

     for(u=inicio+1;u<=inicio+longcad[i]+1;u++)//busca mejor posicion para cada cliente u
     {
        if(cad[u]<n && cad[u]!=-1)
        {
        mejpos[cad[u]]=c1[cad[u]][1];
        for(j=inicio+1;j<=inicio+insert+1;j++)
        {
           if(c1[cad[u]][j]>=mejpos[cad[u]]|| insert==(longcad[i]-1))
           {
              mejpos[cad[u]]=c1[cad[u]][j];
              mpos[cad[u]]=j;
           }
        }
        }
```

```
}
for(u=inicio+1;u<=inicio+longcad[i]+1;u++)//Busca el mejor cliente u segun su mejor posición
{
 if(cad[u]<n && cad[u]!=-1)
 {
   ma1= mejpos[cad[inicio+1]];
   if(mejpos[cad[u]]>=ma1 ||insert==(longcad[i]-1))
  {
     ma1=mejpos[cad[u]];
     maycliente=cad[u];
     bestu=u;

  }
 }
}
for(k=N+m-1;k>=mpos[maycliente];k--)//Recorre todas las casillas 1 posición a la derecha a partir de la nueva inserción
{
   mr[k+1]=mr[k];
   //printf("<3 %d",k);
}mr[mpos[maycliente]]=maycliente;//inserta**********
cad[bestu]=-1;
```

```
      insert++;//*****
      double dem_acum[200],dist_acum[200];
      for(k=3+insert;k>mpos[maycliente];k--)//Actualiza las demandas y distancias acumuladas una vez q se hizo una inserción
      {
        dem_acum[k]=dem_acum[k-1]+dem[maycliente];
        dist_acum[k]=dist_acum[k-1]+mdc[mr[mpos[cad[u]]]-1][maycliente]+mdc[maycliente][mpos[cad[u]]];
      }  dem_acum[mpos[maycliente]]=dem_acum[k-1]-dem[mr[k+1]]+dem[maycliente];
         dist_acum[mpos[maycliente]]=dist_acum[mpos[cad[u]]-1]+mdc[mr[mpos[cad[u]]-1]][maycliente];
      int r;
 }while(insert<longcad[i]);
 inicio=inicio+longcad[i]+2;



}
printf("MEGA RUTA\n");
for(i=0;i<N+m;i++)
   {
       printf("\t");
    printf("%d-",mr[i]);
    //fprintf(programa,"%d-",mr[i]);
```

```
    if(i%10 == 0)
                {    //fputc('\n',programa);
                     }
  }



}



void fcortes()
{



     printf("\n");
printf(" 1 \t ");
inicio=0;

for(i=0;i<N+m;i++)// Inicializa vectores de cemandas tiempos de llegada y tiempos de salida(acum)
{
  demacum_1[i]=0;
  tllegada_1[i]=0;
  tsalida_1[i]=0;
```

```
}
printf("\n\n");
inicio=0;
for(i=0;i<m;i++)//cALCULA TIEMPOS ACUMULADOS Y DEMANDA
{printf("\n\n depot=%d  longcad=%d \n \n ",i,longcad[i]);
 for(j=inicio;j<=inicio+longcad[i]+1;j++)
  {
    if(j==0)
    {   demacum_1[j]=0;
        tllegada_1[j]=0;
        tsalida_1[j]=0;
    }
    else{
           if(mr[j-1]>=n && mr[j]>=n)
            {printf("\n mr[%d]=%d",j,mr[j]);
             demacum_1[j]=0;//d[j] es la demanda del cliente que está en la posición i de la mega ruta;
             tllegada_1[j]=0;
             tsalida_1[j]=0;
             printf("\n j=%d dem1=%d Tllegada1=%f
tsalida1=%f\n",j,demacum_1[j],tllegada_1[j],tsalida_1[j]);
             }else
             {printf("\n mr[%d]=%d",j,mr[j]);
```

```
            demacum_1[j]=demacum_1[j-1]+d[mr[j]];//d[j] es la demanda del cliente que está en la posición i de la mega ruta;

            tllegada_1[j]=tsalida_1[j-1]+mdc[mr[j-1]][mr[j]];

            tsalida_1[j]=tllegada_1[j]+st[mr[j]];

            printf("\n j=%d dem1=%d Tllegada1=%f tsalida1=%f st[mr[j]]=%d\n",j,demacum_1[j],tllegada_1[j],tsalida_1[j],st[mr[j]]);

            }

        }

   }




 inicio=inicio+longcad[i]+2;

}


system("Pause");

printf("\n\n 4 \n ");




l=0;

int Q=200, nr[m],distmax=500, ch=distmax/m; // capacidad del vehículo k

int multatw[100]; // replica de mr pero en rutas
```

```
float costo_llegada[100];

double salida=0,truta[3*m],ttotal, Ctotal;




printf("\n\n");

for(i=0;i<m;i++)

{nr[i]=0;}


inicio=0;


k=0;


for(i=0;i<m;i++)

{

    nr[i]=0;

    vr[k]=i+n;

   printf("vr[%d]=%d\n",k,vr[k]);

   j=inicio+1;

   /******************/

   salida=horai[mr[j]]-mdc[vr[k]][mr[j]];

   printf("Asalida=%f  mr[j]=%d\n",salida,mr[j]);
```

```
/*****************/

demacum_2[k]=0;

tllegada_2[k]=salida;

tsalida_2[k]=salida;

k++;



 while(j<=inicio+longcad[i])

{

  if(demacum_2[k-1]+d[mr[j]]<Q && tsalida_2[k-1]+mdc[vr[k-1]][mr[j]]<salida+distmax+ch)

  {printf("si cabe\t");

        vr[k]=mr[j];

        printf("vr[%d]=%d\n",k,vr[k]);

        demacum_2[k]=demacum_2[k-1]+d[mr[j]];

        tllegada_2[k]=tllegada_2[k-1]+mdc[vr[k-1]][mr[j]];

        tsalida_2[k]=tllegada_2[k]+st[mr[j]];

         costo_llegada[k]=tllegada_2[cad[j]];//CHECAR Y CORREGIR CON RESPECTO A LOS OTROS



        if(tllegada_2[k]<=horai[mr[j]])

        { // printf(" 8 \t ");
```

```
        costo_llegada[k]=tllegada_2[k]+M*(horai[mr[j]]-tllegada_1[k]);
         //multatw[k]=1;
      }
      else{//printf(" 9 \t ");
         if(horaf[mr[j]]<=tllegada_2[k])
         {   //printf(" 10 \t ");

             costo_llegada[k]=tsalida_2[k]+M*(tsalida_2[mr[j]]-horaf[k]);
             //multatw[k]=1;
         }
        }

 j++;
 k++;
  } // fin if(demacum)
else{

vr[k]=i+n;
printf("vr[%d]=%d\t",k,vr[k]);
nr[i]++;
printf("nr[%d]=%d\n",i,nr[i]);
multatw[k]=0;
```

```
demacum_2[k]=demacum_2[k-1];

tllegada_2[k]=tllegada_2[k-1]+mdc[vr[k-1]][vr[k]];//No estoy considerando la distancia de regreso
al depósito

tsalida_2[k]=tllegada_1[k]+st[i+n];


truta[nr[i]]=tsalida_2[k]-salida;

ttotal=ttotal+truta[nr[i]];

Ctotal= Ctotal+(costo_llegada[k-1]-salida);




r[l]=k;

printf("\n otra ruta \n");

k++;

vr[k]=i+n;

printf("vr[%d]=%d\n",k,vr[k]);

   /******************/

salida=horai[mr[j+1]]-mdc[vr[k]][mr[j+1]];

if(salida<0){salida=0;}

printf("Bsalida=%f   mr[j+1]=%d\n",salida,mr[j+1]);


/*****************/

demacum_2[k]=0;
```

```
    tllegada_2[k]=salida;

    tsalida_2[k]=salida;


      k++;

    }//fin del else




    }// fin while


       vr[k]=i+n;

    printf("vr[%d]=%d\t",k,vr[k]);

    nr[i]++;

    printf("nr[%d]=%d\n",i,nr[i]);

    multatw[k]=0;

    demacum_2[k]=demacum_2[k-1];

    tllegada_2[k]=tllegada_2[k-1]+mdc[vr[k-1]][vr[k]];//No estoy considerando la distancia de regreso
al depósito

    tsalida_2[k]=tllegada_1[k]+st[i+n];

    r[l]=k;

    printf("\n otra ruta \n");

    k++;

inicio=inicio+longcad[i]+2;
```

```
printf("\n inicio=%d, longcad=%d\n",inicio,longcad[i]);


}// fin for(i)



int longvr=k;

printf(" \n12 \t ");

system("Pause");

//printf("\n nr=%d\t",nr);

/*for(i=0;i<=k;i++)

{


//printf("vr[i]\t",%d );

printf("\n v[%d]=%d\t",i,vr[i]);

//printf("\n");

printf("multatw[%d]=%d\t",i,multatw[i]);

}*/



/************

calcular nuevas multas para vr[]

con salida=0 y salida =horai-mdc[][]
```

```
y comparar

-- for-

-- activar el el vector multa =1ò 0*/

int sumamultas=0;

printf("\n");

for(i=0;i<=longvr;i++)

{

  if(vr[i]<n)

  {

    if(tllegada_2[i]<horai[i]||horaf[i]<tllegada_2[k])

    {multatw[i]=1;

     sumamultas=sumamultas+1;}

    else{multatw[i]=0;}

  }

  printf("vr[%d]=%d multatw[%d]=%d\t\t",i,vr[i],i,multatw[i]);

} printf("\n\nsumamultas=%d\n ttotal=%lf   Ctotal=%lf\n",sumamultas,ttotal,Ctotal);

}



/************MEJORA***********/

void fmejora()

{
```

```
for(int z=1;z<longcad[i];z++) //¿inicio?
    {
        quitoA[vr[z]]=mdc[vr[z-1]][vr[z]];
        quitoB[vr[z]]=mdc[vr[z]][vr[z+1]];
        quito[vr[z]]=quitoA[vr[z]]+quitoB[vr[z]];

        for(int j=1;j<longcad[i]-1;j++) //Meter un if de validacion, longcad se cambia por longvr[#ruta]
        {
         quitoA[vr[j]]=mdc[vr[j-1]][vr[j]];
         quitoB[vr[j]]=mdc[vr[j]][vr[j+1]];
         quito[vr[j]]=quitoA[vr[j]]+quitoB[vr[j]];

         pongoA[vr[z]]=mdc[vr[j-1]][vr[z]];
         pongoB[vr[z]]=mdc[vr[z]][vr[j+1]];
         pongo[vr[z]]=pongoA[vr[z]]+pongoB[vr[z]];

         pongoA[vr[j]]=mdc[vr[z-1]][vr[j]];
         pongoB[vr[j]]=mdc[vr[j]][vr[z+1]];
         pongo[vr[j]]=pongoA[vr[j]]+pongoB[vr[j]];

         tllegada_3[z]=tllegada_3[z]-quitoA[vr[z]]+pongoA[vr[j]];
```

```
costo_llegada_3[z]=tllegada_3[z];
tllegada_3[j]=tllegada_3[j]-quitoA[vr[j]]+pongoA[vr[z]];
costo_llegada_3[j]=tllegada_3[j];



if(tllegada_3[z]<horai[vr[j]])
{costo_llegada_3[z]=tllegada_3[z]+M*(horai[vr[j]]-tllegada_3[z]);}
  else{if(tllegada_3[z]>horaf[vr[j]])
       {costo_llegada_3[z]=tllegada_3[z]+M*(tllegada_3[z]-horaf[vr[j]]);
     }
}

if(tllegada_3[j]<horai[vr[z]])
{costo_llegada_3[j]=tllegada_3[j]+M*(horai[vr[z]]-tllegada_3[j]);}
  else{if(tllegada_3[j]>horaf[vr[z]])
       {costo_llegada_3[j]=tllegada_3[j]+M*(tllegada_3[j]-horaf[vr[z]]);
     }
}



// ttotal_aux[#ruta]=ttotal[#ruta]-(quito[vr[z]]+quito[vr[j]])+(pongo[vr[z]]+pongo[vr[j]]);
```

```
if(z<j){ru=z; s=j;}else{ru=j; s=z;}
int l;
for(l=ru+1;l<s-1;l++)//for entre los que cambiè
{
   tllegada_3[l]=tllegada_3[l-1]+mdc[vr[l-1]][vr[l]];
   costo_llegada_3[l]=tllegada_3[l];
   /* TW*/
   if(tllegada_3[l]<horai[vr[l]])
   {costo_llegada_3[l]=tllegada_3[l]+M*(horai[vr[l]]-tllegada_3[l]);}
     else{if(tllegada_3[l]>horaf[vr[l]])
        {costo_llegada_3[l]=tllegada_3[l]+M*(tllegada_3[l]-horaf[vr[l]]);
   }//FALTA AGREGAR PARA ACUMULAR LOS COSTOS_LLEGADA,OSEA CON MULTAS
}

tllegada_3[s]=tllegada_3[s-1]+mdc[vr[s-1]][vr[s]];
costo_llegada_3[s]=tllegada_3[s];
//for(l=s+1;l<longvr[ruta];l++)//for entre los que cambiè
for(l=s+1;l<ruta;l++)//for entre los que cambiè
{
   tllegada_3[l]=tllegada_3[l-1]+mdc[vr[l-1]][vr[l]];
   costo_llegada_3[l]=tllegada_3[l];
   /* TW*/
```

```
            if(tllegada_3[l]<horai[vr[l]])
            {costo_llegada_3[l]=tllegada_3[l]+M*(horai[vr[l]]-tllegada_3[l]);}
              else{if(tllegada_3[l]>horaf[vr[l]])
                  {costo_llegada_3[l]=tllegada_3[l]+M*(tllegada_3[l]-horaf[vr[l]]);
            }
          }

           if(vr[i]<n)
           {
                    if(tllegada_3[i]<horai[i]||horaf[i]<tllegada_3[k])
                    {multatw[i]=1;
                     sumamultas[ins]=sumamultas[ins]+1;}
                     else{multatw[i]=0;}
           }
             printf("vr[%d]=%d multatw[%d]=%d\t\t",ins,vr[ins],i,multatw[i]);
           printf("\n\n                    sumamultas[%d]=%d\n                    ttotal[%d]=%lf
Ctotal[%d]=%lf\n",ins,sumamultas[ins],ins,ttotal[ins],ins,Ctotal[ins]);

          printf("\n",costo_llegada_3[l]);

          }//***Ctotal_aux[#ruta]=Ctotal[#ruta]-----FALTA***//
      }}
```