Tesis para obtener el grado de
maestría en ingeniería

Planeación de rutas de vehículos en sistemas con flota privada y pública

Ángela María Toro Hincapié

Febrero de 2013

Universidad EAFIT
Escuela de Ingeniería
Departamento de Ingeniería de Producción

Agradecimientos

Agradezco a mi madre, a mi hermano y a mi familia por siempre brindarme apoyo incondicional y fuerzas para continuar. Por enseñarme que con esfuerzo, perseverancia, paciencia y dedicación se logra alcanzar cualquier objetivo.

También agradezco a mi asesor Mario Cesar Velez, quien me ayudo incansablemente y creyó en mis capacidades y a mi profesor de pregrado Juan Guillermo Villegas, quien me apasiono con el tema de investigación y me inspiro a comenzar este difícil pero satisfactorio viaje por una maestría.

Contenido

Resumen

En este proyecto se propone un método heurístico de solución para el problema de planeación de rutas de vehículos en sistemas con flota privada y pública, conocido en la literatura como *vehicle routing problem with private fleet and common carrier* (VRPPC) Este problema de optimización combinatoria consiste en diseñar una serie de rutas donde cada cliente debe ser atendido una vez, ya sea por un vehículo de la flota privada o asignado a una empresa de transporte público de manera que los costos totales de operación sean mínimos. Para su solución se propone un heurístico de búsqueda de vecindario variable, conocido en la literatura como VNS, que parte de una solución inicial para luego mejorarla por medio de la exploración sistemática de múltiples vecindarios. Para evaluar el desempeño del heurístico propuesto se compararon los resultados obtenidos con los de otros métodos disponibles en la literatura. Los experimentos computacionales demuestran que el heurístico propuesto es efectivo en encontrar soluciones de buena calidad a un costo computacional razonable.

Palabras claves: Programación de Rutas de vehículos (VRP), Flota privada, flota publica, Metaheuristico, Búsqueda en vecindario variable (VNS).

1. Introducción

Actualmente cerca del 50% de la población mundial vive en áreas urbanas, con una tasa de crecimiento urbano del 2% anual [1], [2], por lo tanto surgen estrechas relaciones entre las necesidades básicas humanas como agua, alimento y transporte, y la manera como se suplen. Para satisfacer la demanda creciente de más y mejores servicios es necesario desarrollar herramientas para apoyar la toma de decisiones y la gestión de los recursos. La administración de la cadena de suministro es una de estas herramientas importantes para minimizar la incertidumbre en el horizonte de decisión de las empresas y maximizar el valor total agregado a los productos o servicios que se ofrecen [3]. Los Estados Unidos tienen uno de los sistemas de transporte de mercancías más grande del mundo y sirve como referente para algunas cifras relevantes: en 2008 las compañías de los Estados Unidos recibieron US$22 mil millones por servicios comerciales de carga prestados a empresas de otros países, sus empresas pagaron US$45 mil millones a compañías extranjeras por el transporte de mercancías y US$27 mil millones a puertos extranjeros por los servicios portuarios [4]. En Colombia estudios recientes [5] revelan que la actividad económica de transporte, almacenamiento y comunicación es una de las tres actividades que más contribuye al crecimiento del PIB en el país, con un 3.99% entre los años 2007 y 2008; así mismo concluyen que el costo variable de operación por kilometro para un vehículo de carga se incrementó en un 15% y los costos fijos en un 8% entre 2004 y 2006 [6]. Por lo tanto, el movimiento de carga a nivel mundial muestra una tendencia ascendente, convirtiendo el transporte en una de las decisiones operativas más importantes en la cadena de suministro.

Dentro de las operaciones de transporte una de las decisiones que debe tomarse con frecuencia es el diseño de rutas de vehículos. En las industrias productoras de bienes para atender la demanda de los centros de distribución, bodegas, almacenes y clientes finales, y en las empresas prestadoras de servicios para atender demandas de servicios públicos domiciliarios, como la recolección de basura, mensajería, entrega de diarios, servicio de transporte masivo, entre otros. El diseño de rutas de vehículos ha sido adaptado y utilizado en diversas industrias obteniendo ahorros de entre el 7% y el 37% de algunos de los costos asociados [7]. En aplicaciones reales el uso de técnicas de programación matemática y herramientas computacionales para la planeación del proceso de distribución y las operaciones de transporte ha producido importantes ahorros, generalmente entre el 5% y el 20% del costo global de transporte. Además, el proceso de transporte es transversal a todas las etapas del sistema de producción y distribución y representa un componente importante del costo de los productos o servicios, oscilando generalmente entre el 10% y el 20% del costo total [8].

En la literatura científica el problema del diseño de rutas de vehículos se encuentra bajo el nombre genérico de *vehicle routing problem* (VRP) [9]. El VRP es uno de los problemas de optimización combinatoria más importantes y estudiados en los campos de administración de la cadena de suministro, investigación de operaciones y técnicas de programación matemática [10] y se define como el diseño de una serie de rutas para una flota de vehículos que debe atender un número determinado de clientes, con el objetivo de optimizar una función objetivo que en la mayoría de los casos es el costo total asociado a las rutas diseñadas [8]. Las variaciones o extensiones más

estudiadas del VRP plantean problemas que incluyen vehículos con capacidades y costos diferentes, horarios prestablecidos de entrega, restricciones en la longitud del recorrido, múltiples bodegas, múltiples funciones objetivo, demanda u otros componentes estocásticos, disponibilidad de flotas privadas y públicas, entre otros. Cuatro importantes variaciones del VRP en la literatura son (i) el problema de ruteo de vehículos con capacidad limitada o finita (CVRP), (ii) el problema de ruteo de vehículos con ventanas de tiempo (VRPTW), donde cada cliente debe ser atendido dentro de un intervalo de tiempo determinado; (iii) el problema de ruteo de vehículos con viajes de regreso (VRPB) y (iv) el problema de ruteo de vehículos con recogida y entrega (VRPPD); aplicadas en problemas reales de recolección de basura, entrega y recolección de alimentos y en la industria de diarios, etc. [11]. Las principales técnicas propuestas para la solución de este tipo de problemas se pueden agrupar en dos métodos: exactos y de aproximación o heurísticos [12]. Según se pudo constatar en la revisión de la literatura, las instancias más grandes que han sido resueltas óptimamente se han resuelto para el CVRP y contienen alrededor de 135 clientes [13]. Instancias más grandes solo pueden ser resueltas óptimamente en casos aislados. Por lo tanto, debido a que el VRP y sus variaciones son problemas computacionalmente complejos (*NP-Hard*) [14], los métodos heurísticos han jugado un papel preponderante en el desarrollo de métodos de solución.

Una variación del CVRP que ha tenido relativa poca atención en la literatura es el VRP con flota privada y flota pública, conocido en la literatura como *vehicle routing problem with private fleet and common carrier* (VRPPC). En el VRPPC el objetivo es diseñar una serie de rutas donde cada cliente debe ser atendido una vez, ya sea por un vehículo de la flota privada o asignado a una empresa de transporte público de manera que los costos totales de operación sean mínimos [15–18]. Este problema particular es relevante cuando las compañías buscan atender todas sus órdenes y maximizar el uso de su flota privada, pero los costos fijos y de operación de esta son cada vez más altos y la programación de las rutas rudimentaria, por lo tanto se desarrolla la tendencia a reducir el número de vehículos de la flota privada y atender órdenes con una flota externa para reducir costos operativos. Recientes estudios muestran que solo un 30% de las demandas son atendidas por la flota privada de las empresas [19]; todo esto debido a la falta de una planeación operacional integrada que determine el balance y minimice el costo.

En esta investigación se propone un método heurístico para resolver el VRPPC basado en los métodos de búsqueda en vecindario variable o *variable neighborhood search* (VNS) [20] que han sido utilizados con éxito para resolver problemas de optimización combinatoria [21]. Este documento está organizado de la siguiente manera: en la sección 2 se presenta la revisión de la literatura sobre el VRPPC y otros problemas relacionados. La sección 3 presenta la formulación matemática del problema como un problema lineal entero mixto. La sección presenta 4 un resumen de los resultados computacionales obtenidos al implementar el modelo matemático en un software comercial de optimización sobre un conjunto de instancias de prueba disponibles en la literatura. En la sección 5 se presenta el método heurístico de solución propuesto, y en la sección 6 se presentan los resultados computacionales obtenidos al implementar el heurístico propuesto sobre el mismo conjunto de instancias de prueba usadas para evaluar el modelo matemático. En la sección 7 se presentan las conclusiones del proyecto así como posible extensiones y futuras líneas de investigación.

2. Revisión de la Literatura

El VRP fue introducido en 1959 por Dantzig y Ramser [22] al describir una aplicación real que consistió en encontrar las rutas óptimas para atender un sistema de estaciones de gasolina desde una terminal con una flota de camiones de entrega. Los autores propusieron una formulación matemática del problema como una generalización del problema del agente viajero o *traveling salesman problem* (TSP) así como un algoritmo de aproximación. En 1964 Clarke y Wright [23] propusieron el heurístico que lleva el nombre de sus autores (C&W) y que mejoró en su momento la aproximación propuesta en [22]. Posteriormente gran cantidad de modelos y algoritmos han sido propuestos para resolver el VRP y sus diferentes variaciones [10]. Dentro de las variaciones más comunes se encuentra el CVRP [8], [14], [24], el cual se define como el diseño de una serie de rutas para atender un número de clientes con demanda y localización conocida, con una flota de vehículos con restricciones de capacidad, donde cada cliente es visitado una sola vez por un solo vehículo tal que se minimice la suma de los costos. Un número importante de variaciones o extensiones para el VRP han sido estudiadas debido a que el CVRP original no tiene en cuenta algunas restricciones y características de los problemas reales tales como flota de vehículos con capacidades y costos diferentes (flota heterogénea) [25], horarios de entrega o recepción prestablecidos [26], múltiples bodegas [27], múltiples funciones objetivo [28], demanda u otros componentes estocásticos [29], entre otros.

Los métodos de solución para el VRP propuestos por la comunidad de investigación de operaciones se pueden clasificar en tres grupos principales: el primer grupo corresponde a los métodos exactos [30][31], que utilizan formulaciones de programación matemática heredadas en su gran mayoría del TSP, diseñadas para encontrar soluciones óptimas como *branch-and-bound* y algoritmos basados en relajación *lagrangiana* entre otros [32]. El segundo grupo corresponde a los métodos heurísticos [33][12], compuestos por métodos de construcción inicial como el heurístico C&W [23][34], métodos de dos fases como el algoritmo *Sweep* [35] y heurísticos de mejoramiento con búsqueda local como el λ-opt [36]. En el tercer grupo están los métodos metaheurísticos [37][38] en los que se integran algunos métodos de construcción con métodos heurísticos de mejoramiento, en los cuales se busca explorar un conjunto de soluciones factibles y escapar de óptimos locales por medio de alguna estrategia como *simulated annealing* (SA) [39], *tabu search* (TS) [25], [40], *ant colony optimization* (ACO) [28], [41] y algoritmos genéticos (GA) [42], entre otros. Normalmente, los métodos clásicos, exactos y heurísticos, como el C&W, *Sweep*, *branch-and-bound*, entre otros han obtenido soluciones entre el 2% y el 10% por encima de la mejor solución conocida, mientras que la mejor implementación de métodos metaheurísticos arroja soluciones que por lo general difieren de la mejor solución conocida en un 0.5% o menos [8].

Un caso particular de variación del VRP clásico es el VRP con flota de transporte privada y pública, conocido en la literatura como el *vehicle routing problem with private fleet and common carrier* (VRPPC), donde la decisión del modo de transporte depende de varios factores como la demanda y localización de los clientes, las características del producto, los objetivos del servicio, la red física de transporte disponible, el tipo de flota (homogénea o heterogénea), los tipos de compañías que prestan el servicio y los costos, entre otros. En 1987 Volgenant et al. [17] formularon por primera

vez este problema con un solo vehículo como un TSP. En 1995 Diaby et al. [43] lo resolvieron óptimamente con el algoritmo *branch-and-bound* para problemas con hasta 200 clientes. Hasta donde se pudo constatar en la literatura Chu [44] planteó por primera vez esta variación del VRP con más de un vehículo y la resolvió con el heurístico denominado *truck load and less-than truck load carrier* (TL-LTL), haciendo una selección inicial de los clientes que serán atendidos por la flota externa según el costo del servicio, seguida del heurístico de construcción inicial C&W y finalmente mejorando el resultado con métodos de búsqueda local. En [16] Bolduc et al. mejoraron los resultados de Chu con el heurístico denominado *selection, routing and improvement* (SRI), usando el heurístico C&W para construir dos soluciones iniciales diferentes y mejorarlas con intercambios entre clientes más sofisticados como el 4-opt. En [45] Bolduc et al. propusieron el metaheuristico denominado *randomized construction improvement-perturbation* (RIP), que usa C&W para hacer la construcción inicial con una modificación al procedimiento de construcción, aleatorizando el cálculo de los ahorros en cada fase y finalmente implementando varios procedimientos de mejora. En [17] Côté et al. describen una nueva aproximación metaheuristica solo para flota homogénea, basada en *tabu search* (TS), en la cual utilizan el heurístico de inserción de mínimo costo (*least cost insertion*) para la creación de las rutas iniciales y luego un heurístico de búsqueda local para mejorar los resultados obtenidos por el RIP. En [46] Potvin et al. se propone un heurístico basado en *tabu search* (TS+) que utilizan dos heurísticos aleatorizados para obtener una solución inicial (*least-cost insertion y convex hull insertion*), para luego mejorar la solución con una estructura de vecindario basada en cadenas de eyección. En [47] Euvhi et al. describen el heurístico denominado *iterated density estimation evolutionary algorithm* (IDEA), en el cual se explora el espacio de búsqueda mediante muestreo aleatorio usando una distribución de probabilidad que se desarrolla durante la búsqueda. Finalmente en [42] Kratica et al. se propone un algoritmo genético (GA) en el cual por medio de una técnica de búsqueda estocástica se imitan algunos procesos espontáneos de optimización en la selección natural y la reproducción; fue probado con las instancias pequeñas propuestas en [44] alcanzando las soluciones óptimas para 7 de las 10 instancias y es estrictamente mejor que el SRI y el TL-LTL para tres casos. Para probar el algoritmo con las instancias grandes propuestas en [45] se incluyó en el algoritmo la búsqueda local basada en cadenas de eyección propuesta en [46], pero aumentaba considerablemente el tiempo computacional sin obtener mejores soluciones.

En [26] [48] se proponen dos problemas relacionados al VRPPC, donde se determina al mismo tiempo el tamaño de la flota heterogénea y se determinan las rutas para los vehículos, con el fin de servir a un número predeterminado de clientes con demandas conocidas desde un almacén central. Para estos problemas también se dispone de un número determinado de tipos de vehículos con costos fijos y capacidades conocidas pero de cada tipo de vehículo se asume que hay disponibles un número infinito. En [49] se presenta una metodología para determinar si un cliente debe ser atendido por la flota privada y una ruta de entrega o por una empresa de transporte público, por medio de una interacción de los costos a través del uso de una aproximación de la raíz cuadrada de la longitud de la ruta. Se presentan tres técnicas analíticas donde se busca optimizar el tamaño de la flota y la frecuencia del servicio según sean estas variables o fijas. Finalmente en Stenger et al. [21] se propone el algoritmo denominado *adaptative variable neighbourhood search* (AVNS) para resolver el VRPPC con múltiples depósitos o *multi depot vehicle routing problem with private fleet and common carrier* (MDVRPPC), una extensión del VRPPC en la que se consideran múltiples

depósitos y múltiples flotas externas o subcontratistas. El AVNS propuesto también se utilizó para resolver el VRPPC y el VRP con varios depósitos (MDVRP) mejorando considerablemente la calidad de las soluciones para todos los problemas estudiados.

Hasta donde se pudo constatar en la revisión de la literatura, los trabajos académicos más relevantes alrededor del problema denominado VRPPC son los trabajos de las referencias [17], [21], [42], [45], [46]. Para el desarrollo de este proyecto se tomarán como referencia los resultados reportados en estos artículos para evaluar el desempeño de los métodos de solución aquí propuestos.

3. Modelo Matemático

En la literatura se han propuesto tres modelos básicos para el resolver el VRP [8]. Una formulación basada en el flujo de los vehículos, una formulación basada en el flujo de los productos y una formulación basada en el problema clásico de partición de conjuntos (*set partitioning problem o* SPP). Todas estas formulaciones pueden ser extendidas o adaptadas para resolver las diferentes variantes del VRP incorporando restricciones y variables, o cambiando la función objetivo. El modelo más frecuentemente usado como base para formular variantes del VRP es el basado en flujo de vehículos en el cual se definen variables enteras asociadas a cada arco del grafo que cuentan el número de veces que el arco es atravesado por un vehículo. En esta formulación el costo de la solución se puede expresar como la suma de los costos asociados con los arcos, y las restricciones más relevantes tienen que ver con la transición de los clientes dentro de las rutas.

Para modelar el VRPPC como un problema de programación entera mixta, se presenta a continuación la formulación basada en flujo de vehículos propuesta en [45]. Sea $G = (V, A)$ un grafo dirigido donde $V = \{0, \dots . n\}$ es el conjunto de vértices (nodos) y $A = \{(i, j): i, j \in V, i \neq j\}$ es el conjunto de arcos, cuyos elementos son pares ordenados de vértices diferentes. El vértice 0 es el depósito o almacén y los demás vértices representan los clientes que deben ser visitados. Cada cliente $i \in V\backslash\{0\}$ está caracterizado por una demanda q_i y por un costo fijo e_i si es atendido por la flota externa, independiente de la secuencia de la ruta. Sea $M = \{1, \dots, m\}$ el conjunto de *m* vehículos que conforman la flota privada, inicialmente localizados en el depósito. Cada vehículo $k \in M$ tiene capacidad Q_k y se incurre en un costo fijo f_k cada vez que un vehículo de la flota privada es utilizado. La matriz de costo de viaje C_{ijk} se define en $A \times M$.

El VRPPC consiste en servir a todos los clientes de tal manera que: (1) cada cliente sea visitado exactamente una vez ya sea por la flota privada o por la flota pública, (2) cada vehículo de la flota privada sirve a una sola ruta que comienza y termina en el depósito, (3) la demanda total de cualquier ruta no debe exceder la capacidad del vehículo asignado a esta, y (4) la suma del costo fijo de los vehículos usados, el costo de los viajes y el costo de la flota externa sea mínimo. Para modelar las diferentes decisiones que se abordan en esta formulación del VRPPC se utilizan cuatro conjuntos de variables de decisión:

X_{ijk}: Variable binaria que indica la secuencia en la que cada vehículo visita a los clientes. La variable X_{ijk} toma el valor de uno si el vehículo *k* visita el vértice *j* inmediatamente después de visitar el vértice *i*, y cero de lo contrario.
Y_{ik}: Variable binaria que indica qué vehículo visita cada cliente. La variable Y_{ik} toma el valor de uno si el vehículo *k* visita el cliente *i*, y cero de lo contrario.
Z_i: Variable binaria que indica qué clientes son asignados a la flota externa. La variable Z_i toma el valor de uno si el cliente *i* es atendido por la flota externa, y cero de lo contrario.
U_{ik}: Límite superior de la carga del vehículo *k* al dejar el cliente *i*.

Usando la notación anterior la formulación del VRPPC es la siguiente:

$$Min \sum_{k=1}^{m} f_k\, y_{0k} + \sum_{i=0}^{n} \sum_{\substack{j=0 \\ j\neq i}}^{n} \sum_{k=1}^{m} C_{ijk} X_{ijk} + \sum_{i=1}^{n} e_i\, Z_i \qquad 3.1$$

Sujeto a:

$$\sum_{j=1}^{n} \sum_{k=1}^{m} X_{0jk} = \sum_{i=1}^{n} \sum_{k=1}^{m} X_{i0k} \leq m \qquad 3.2$$

$$\sum_{\substack{j=0 \\ j\neq h}}^{n} X_{hjk} = \sum_{\substack{i=0 \\ i\neq h}}^{n} X_{ihk} = y_{hk} \qquad \forall h\epsilon V;\ \forall k\epsilon M \qquad 3.3$$

$$Z_i + \sum_{k=1}^{m} y_{ik} = 1 \qquad \forall i\ \epsilon\ V\backslash\{0\} \qquad 3.4$$

$$\sum_{i=1}^{n} q_i y_{ik} \leq Q_k \qquad \forall k\epsilon M \qquad 3.5$$

$$u_{ik} - u_{jk} + Q_k X_{ijk} \leq Q_k - q_j \qquad \forall i,j\ \epsilon\ V\backslash\{0\}, \mathrm{i} \neq \mathrm{j};\ k\epsilon M \qquad 3.6$$

$$X_{ijk}\ \epsilon\ \{0,1\} \qquad \forall i,j\epsilon\ V, \mathrm{i} \neq \mathrm{j};\ \mathrm{k}\epsilon\mathrm{M} \qquad 3.7$$

$$y_{ik}\epsilon\ \{0,1\} \qquad \forall i\epsilon V;\ \forall k\epsilon M \qquad 3.8$$

$$Z_i\ \epsilon\ \{0,1\} \qquad \forall i\ \epsilon\ V\backslash\{0\} \qquad 3.9$$

$$u_{ik} \geq 0 \qquad \forall i\epsilon\ V\backslash\{0\};\ \forall \mathrm{k}\epsilon\mathrm{M} \qquad 3.10$$

La función objetivo (3.1) minimiza la suma de los costos fijos de los vehículos usados, los costos de las rutas internas y los costos de la flota externa. El conjunto de restricciones (3.2) asegura que máximo *m* vehículos de la flota privada pueden ser usados en la solución. El conjunto de restricciones (3.3) asegura que el mismo vehículo *k* debe entrar y dejar el cliente *h*. Con el conjunto de restricciones (3.4) se asegura que cada cliente debe ser asignado ya sea a la flota privada o a la flota pública. El conjunto de restricciones (3.5) asegura que no se exceda la capacidad de cada vehículo, mientras que el conjunto de restricciones (3.6) asegura que se eliminen las rutas que no incluyen el depósito. Los ecuaciones 3.7 – 3.10 constituyen las restricciones de dominio de las variables del modelo.

3.1 Instancias de prueba

Para evaluar el desempeño computacional del modelo matemático propuesto en [45] se utilizó un total de 44 instancias de prueba propuestas en la literatura. Inicialmente el modelo se probó en dos conjuntos de diez instancias pequeñas y heterogéneas (i.e. vehículos no idénticos) propuestos por Chu (Chu-H) [44] y por Bolduc et. al (B-H) [16]. En la Tabla 3.1 se presentan las características generales de este primer conjunto de instancias de prueba: las primeras dos columnas corresponden al nombre de la instancia y al número de clientes en la misma. Las columnas restantes describen la flota de vehículos así: Debido a que las instancias pueden tener hasta tres tipos diferentes de

vehículos (i.e. A, B y C), para cada tipo de vehículo se definen lo escalares $m(\cdot)$, $Q(\cdot)$, $f(\cdot)$ y $c(\cdot)$ que representan respectivamente la cantidad de vehículos por tipo, la capacidad, el costo fijo y el costo variable por unidad de distancia asociados al uso de cada tipo de vehículo.

Tabla 3.1. Instancias de prueba con flota heterogénea

Instancia	*n*	*Vehículo tipo A*				*Vehículo tipo B*				*Vehículo tipo C*			
		m(A)	*Q(A)*	*f(A)*	*c(A)*	*m(B)*	*Q(B)*	*f(B)*	*c(B)*	*m(C)*	*Q(C)*	*f(C)*	*c(C)*
Chu-H-01	5	1	40	60	1.5	1	30	50	1.5				
Chu-H-02	10	1	75	120	1.5	1	65	100	1.5				
Chu-H-03	15	1	110	150	1.5	1	100	140	1.5	1	90	130	1.5
Chu-H-04	22	1	4500	250	1.5	1	4000	200	1.5				
Chu-H-05	29	1	4500	250	1.5	1	4000	200	1.5	1	3500	180	1.5
B-H-01	5	1	40	60	1.5	1	30	50	1.5				
B-H-02	10	1	75	120	1.5	1	65	100	1.5				
B-H-03	15	1	110	150	1.5	1	100	140	1.5	1	90	130	1.5
B-H-04	22	1	4500	250	1.5	1	4000	200	1.5				
B-H-05	29	1	4500	250	1.5	1	4000	200	1.5	1	3500	180	1.5

Adicionalmente el modelo matemático se probó en un conjunto de instancias homogéneas (i.e. vehículos idénticos) de mayor tamaño. En este caso se usaron en total las 34 instancias propuestas por Christofides y Eilon (CE) [50] y por Golden et al. (G) [51]. En la Tabla 3.2 se presentan las características generales de este conjunto de instancias homogéneas.

Tabla 3.2. Instancias de prueba con flota homogénea

Instancia	*n*	*m*	*Q*	*f*	*c*
CE-01	50	4	160	120	1.00
CE-02	75	9	140	100	1.00
CE-03	100	6	200	140	1.00
CE-04	150	9	200	120	1.00
CE-05	199	13	200	100	1.00
CE-06	50	4	160	140	1.00
CE-07	75	9	140	120	1.00
CE-08	100	6	200	160	1.00
CE-09	150	10	200	120	1.00
CE-10	199	13	200	120	1.00
CE-11	120	6	200	180	1.00
CE-12	100	8	200	120	1.00
CE-13	120	6	200	260	1.00
CE-14	100	7	200	140	1.00
G-01	240	7	550	820	1.00
G-02	320	8	700	1060	1.00
G-03	400	8	900	1380	1.00
G-04	480	8	1000	1720	1.00
G-05	200	4	900	1620	1.00
G-06	280	5	900	1700	1.00
G-07	360	7	900	1460	1.00
G-08	440	8	900	1480	1.00
G-09	255	11	1000	60	1.00
G-10	323	13	1000	60	1.00
G-11	399	14	1000	80	1.00
G-12	483	15	1000	80	1.00
G-13	252	21	1000	60	1.00
G-14	320	23	1000	60	1.00
G-15	396	26	1000	60	1.00
G-16	480	29	1000	60	1.00
G-17	240	18	200	40	1.00
G-18	300	22	200	60	1.00
G-19	360	26	200	60	1.00
G-20	420	31	200	60	1.00

Los escalares *n*, *m*, *Q*, *f* y *c* corresponden respectivamente al número de clientes a atender, el número de vehículos idénticos que conforman la flota privada, el valor de capacidad de los vehículos, el costo fijo asociado al uso de un vehículo de la flota privada y el costo variable por unidad de distancia asociado a la flota privada. Una descripción detallada sobre como fueron generadas estas instancias se encuentra en [45], y el conjunto completo de instancias de prueba está disponible en [52].

3.2 Resultados computacionales

El modelo de programación entera mixta se implementó en un software comercial de optimización (i.e. CPLEX ® 12.2) y los experimentos se ejecutaron en un equipo de cómputo dotado con 8 procesadores Intel® Core i7 de 3.07 de GHz y 16 GB de memoria RAM bajo ambiente Windows 7® de 64 bits. El modelo implementado fue ejecutado para cada una de las 44 instancias, y se asignó un tiempo máximo de cómputo por instancia de 3600 segundos (una hora). En todos los casos se reportó la mejor cota inferior (MCI) y la mejor solución entera (MSE) encontradas por el software de optimización. Así mismo se reportó el tiempo de cómputo en segundos (CPU) y el margen de optimalidad (%Δ); es decir, la diferencia porcentual entre MCI y MSE, que se calcula como en la ecuación 3.11. Para los casos en que el software de optimización converge a la solución óptima, el valor de %Δ es igual a cero.

$$\%\Delta = \frac{MSE - MCI}{MSE} \times 100\% \qquad 3.11$$

En el caso de las instancias pequeñas y heterogéneas, el software de optimización pudo encontrar la solución óptima para cuatro de las diez instancias evaluadas. Para las seis instancias restantes el margen promedio de optimalidad fue del 43%. En la Tabla 3.3 se presenta un resumen de los resultados obtenidos en este primer conjunto de instancias.

Tabla 3.3. Resumen de los resultados computacionales en instancias heterogéneas

Instancia	*n*	*MCI*	*MSE*	*CPU*	*%Δ*
Chu-H-01	5	387.50	387.50	2.85	0%
Chu-H-02	10	586.00	586.00	11.04	0%
Chu-H-03	15	459.45	834.00	3600	45%
Chu-H-04	22	981.87	1389.00	3600	29%
Chu-H-05	29	631.78	1488.00	3600	58%
B-H-01	5	423.50	423.50	2.53	0%
B-H-02	10	476.50	476.50	13.02	0%
B-H-03	15	418.75	777.00	3600	46%
B-H-04	22	1051.61	1521.00	3600	31%
B-H-05	29	840.16	1621.50	3600	48%

Para el caso de las instancias de mayor tamaño, de las 34 instancias evaluadas sólo fue posible calcular el margen de optimalidad para 22 de ellas. En el resto de los casos este cálculo no fue posible, ya sea porque el software de optimización no logró calcular la relajación lineal del problema para así obtener un primer valor para la MCI, o porque no fue posible obtener una primera solución entera factible durante la hora de cómputo asignada. Para las 22 instancias en las que se pudo calcular el margen de optimalidad, éste fue en promedio del 82.87%. En la Tabla 3.4 se presenta un resumen de los resultados obtenidos. Los resultados anteriores evidencian la dificultad

computacional del problema en estudio. El margen de optimalidad es considerablemente alto, incluso para las instancias más pequeñas con 50 clientes y cuatro vehículos. Bajo estas circunstancias se hace necesario recurrir a métodos de solución heurísticos o de aproximación que, aunque sin garantía de optimalidad, permitan obtener soluciones de calidad aceptable en instancias de tamaño realista en un tiempo computacional razonable.

Tabla 3.4. Resumen de los resultados computacionales en instancias homogéneas

Instancia	*n*	*MCI*	*MSE*	*CPU*	*%Δ*
CE-01	50	521.11	1242	3654.09	58%
CE-02	75	564.09	2637	3695.93	79%
CE-03	100	855.04	3405	3861.54	75%
CE-04	150	971.51	5869	3630.10	83%
CE-05	199	1155.31	7742	3607.05	85%
CE-06	50	527.99	1367	3634.01	61%
CE-07	75	571.27	2875	3675.28	80%
CE-08	100	870.71	3760	3790.59	77%
CE-09	150	738.40	5908	3646.15	88%
CE-10	199	1200.47	7985	3612.59	85%
CE-11	120	578.93	7647	3700.61	92%
CE-12	100	518.83	3998	3639.01	87%
CE-13	120	630.04	7546	3631.74	92%
CE-14	100	798.00	3848	3605.43	79%
G-01	240	7015.00	43560	3604.24	84%
G-02	320	8762.33	73327	3610.01	88%
G-03	400	11175.13	110200	3666.44	90%
G-04	480	17824.00	154320	3887.68	88%
G-05	200	7278.67	55900	4188.03	87%
G-06	280	12050.04	77700	3607.55	84%
G-07	360	-1.00E+75[1]	99180	5074.59	---
G-08	440	15455.40	120780	3643.68	87%
G-09	255	341.19	4906	3655.68	93%
G-10	323	---[2]	---	---	---

[1] En este caso el software de optimización no logró resolver la relajación lineal del problema
[2] De la instancia G-10 en adelanta el software de optimización no logró encontrar una solución entera factible durante el tiempo asignado.

4. Método de Solución

En este capítulo se propone un método de solución basado en el heurístico denominado búsqueda en vecindario variable (VNS)[3] para resolver instancias homogéneas del VRPPC. El VNS es un heurístico de mejoramiento introducido por Mladenovic y Hansen [53] en el cual se explota el hecho de que un óptimo local con respecto a un determinado vecindario puede no ser un óptimo local para otro vecindario. Basados en esta observación el heurístico VNS cambia de vecindario si durante la exploración de un vecindario particular el algoritmo se queda atrapado en un óptimo local, buscando que bajo el nuevo vecindario la solución actual no sea un óptimo local y el proceso de mejoramiento de la solución pueda continuar. Dependiendo de la forma en que se tome la decisión de pasar de un vecindario a otro, se configuran tres variantes básicas del VNS: El VNS descendente (VND), el VNS reducido (RVNS) y el VNS básico (BVNS). Las tres variantes parten del conjunto $N=\{N_1, N_2, \ldots, N_K\}$ de K vecindarios seleccionados a priori. En la variante VND los vecindarios se exploran de manera determinista (i.e. sin aleatoriedad), a través de un procedimiento de búsqueda local que puede ser de dos tipos: *First Improve* (FI) o *Best Improve* (BI). En una búsqueda local FI, el vecindario se explora hasta que se encuentra la primera solución vecina mejor que la solución inicial. Una vez encontrada ésta, se adopta como la nueva mejor solución y el proceso de exploración del vecindario se inicia nuevamente. En una búsqueda BI, se adopta como nueva solución la mejor solución encontrada luego de explorar exhaustivamente el vecindario. La decisión de cambiar de un vecindario al siguiente bajo la variante VND ocurre cuando el procedimiento de búsqueda local queda atrapado en un óptimo local. El algoritmo termina cuando explora los K vecindarios consecutivamente, sin mejorar la solución. En la variante RVNS, en vez de explorar exhaustivamente los vecindarios, éstos se muestrean aleatoriamente. El criterio de parada en este caso puede variar, ya sea porque el algoritmo evalúe una cantidad de soluciones vecinas sin mejorar la solución actual, porque el algoritmo alcance un tiempo de ejecución máximo, o por una combinación de las anteriores. Por último, la variante BVNS es similar a la variante RVNS, en la cual se aplica un procedimiento de búsqueda local cada vez que se muestrea aleatoriamente una solución vecina a la solución actual. Una explicación detallada del heurístico VNS y sus diferentes variantes puede encontrarse en la referencia [20].

El método de solución propuesto para resolver el problema en estudio es un heurístico VND bajo un procedimiento de búsqueda local de tipo BI, que parte de una solución inicial para luego mejorarla por medio de la exploración sistemática de múltiples vecindarios. Al finalizar el proceso de búsqueda el algoritmo reporta la mejor solución encontrada y el tiempo que tardó en encontrarla. Para mejorar el desempeño del algoritmo se propone ejecutarlo múltiples veces, cada una con una solución inicial diferente, y almacenar la mejor solución encontrada por el heurístico durante las múltiples ejecuciones. El resto de este capítulo está estructurado de la siguiente manera: primero se describe la forma como se construyen las soluciones iniciales, luego se describen los diferentes vecindarios que explora el heurístico y, por último, se describe el algoritmo propuesto.

[3] La búsqueda en vecindario variable se conoce en la literatura científica como *variable neighborhood search* o VNS.

4.1 Notación

Además de la notación descrita en el capítulo 3 de este documento, en este capítulo se agregan los siguientes elementos. Sea $S = \{R, C\}$ una solución para una instancia arbitraria del VRPPC con flota homogénea, donde $R = \{r_1, r_2, \dots\}$ es el conjunto de rutas a atender por la flota privada, y cada ruta $r_j = \{l_0, l_1, l_2 \dots, l_0\}$ inicia y termina en el depósito (i.e. el vértice $l_0 \epsilon\, V$). La ruta $r_j \epsilon\, R$ es un conjunto ordenado de vértices que determina el orden en que el vehículo asignado a la ruta recorre los puntos en que están ubicados los clientes, y el conjunto $C = \{v_1, v_2, \dots\}$ es el conjunto ordenado de vértices asignados a la flota pública, tal que $R \cup C = V$ y $R \cap C = \emptyset$. Sea η_j el número de clientes visitados por la ruta r_j, y sea $r_j(k)$ el *k*-ésimo cliente visitado por la ruta r_j. Sea $g(r_j)$ el costo de atender la ruta $r_j \epsilon\, R$ con la flota privada, y sea $h(r_j)$ el costo de asignar todos los clientes de la ruta r_j a la flota pública. Sean $d(r_j)$ y $q(r_j)$ la distancia total y la carga asociadas a la ruta $r_j \epsilon\, R$ respectivamente. Se definen $G(R)$ y $H(C)$ como los costos de atender los clientes asignados a la flota privada y a la flota pública respectivamente (i.e. los clientes asignados a los conjunto *R* y *C*), tal que $G(R) = \sum_{r_j \in R} g(r_j)$ y $H(C) = \sum_{i \in C} e_i$.

4.2 Soluciones iniciales

En esta sección se presenta inicialmente la forma en que el heurístico C&W encuentra una solución factible para una instancia arbitraria del CVRP con flota homogénea y sin limitación en el tamaño de la flota ($m=\infty$). Posteriormente se propone una variación al C&W que permite encontrar una solución factible inicial para una instancia arbitraria del VRPPC.

Heurístico C&W

El heurístico C&W parte de una solución inicial, la cual es generalmente un conjunto de *n* rutas, cada una con un único cliente como destino. Durante el desarrollo del heurístico se evalúan todos los posibles pares de rutas r_j y r_k en la solución actual con el fin de evaluar la conveniencia de unirlas en una sola y crear la ruta r_{j+k}. Para que se de la unión de las rutas r_j y r_k se deben cumplir las siguientes tres condiciones:

(1) No se viola la restricción de capacidad del vehículo: $q(r_{j+k}) \leq Q$

(2) Se obtiene un ahorro en cuanto a la distancia recorrida: $d(r_{j+k}) < d(r_j) + d(r_k)$

(3) No se cambia el orden en que se visitan los clientes, es decir, el orden o secuencia en el que se visitan los clientes en las rutas r_j y r_k se mantiene en la ruta resultante r_{j+k}.

Para obtener el ahorro potencial de unir dos rutas que señala la condición (2) se pre-calcula $S_{s,t} = C_{0,s} + C_{t,0} - \lambda C_{s,t}$ para cada par de vértices $(s,t)\, \epsilon\, V \times V$. El factor λ es una variable aleatoria uniforme generada en el intervalo 1±ε, donde ε es un parámetro del algoritmo. Este factor, que no está incluido en el heurístico original, fue propuesto inicialmente en [54] y [55] para mejorar el desempeño del mismo. La condición (3) implica que las rutas r_j y r_k se pueden unir para formar la ruta r_{j+k} solamente si el último cliente visitado en r_j se une con el primer cliente visitado en r_k, como se muestra en la Figura 4.1.

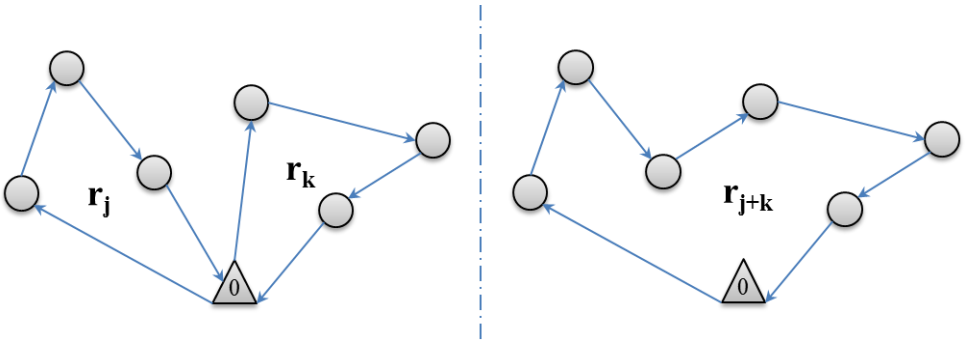


Figura 4.1. Ejemplo heurístico C&W

El heurístico C&W considera todos los posibles pares de rutas y une las dos rutas que cumpliendo con las condiciones anteriormente señaladas genera el mayor ahorro potencial. El proceso continúa hasta que ningún par de rutas cumpla con las condiciones descritas. En el seudocódigo del heurístico C&W que se describe a continuación se devuelve en *R* un conjunto de rutas factibles que contienen a todos los clientes de la instancia.

```
R ← CW(G, A, Q, ε)
   //Inicialización
   R ← {ϕ}
   for i = 1 to n
      r_i ← {0, l_i, 0}
      R ← R U {r_i}
   next i
   //Cálculo de ahorros
   for j = 1 to n-1
      for k = j+1 to n
       Generar λ ~ U(1-ε, 1+ε)
       S_j,k ← C_0,j + C_k,0 − λC_j,k
      next k
   next j
   //Unir rutas
   (i*,j*) ← argmax{S_i,j}
   while S_i*,j* > 0
      r_i* ← {k|i* ∈ r_k}, r_j* ← {k|j* ∈ r_k}
      if r_i* ≠ r_j* then
         if q(r_i*) + q(r_j*) ≤ Q then
            if i* = r_i*(η_i*) and j* = r_j*(1) then
               R ← R U {r_i+j}, R ← R \ {r_i*}, R ← R \ {r_j*}
            end if
            if j* = r_j*(η_j*) and i* = r_i*(1) then
               R ← R U {r_j+i}, R ← R \ {r_i*}, R ← R \ {r_j*}
            end if
         end if
      end if
      S_i*,j* ← -∞, (i*,j*) ← argmax{S_i,j}
   repeat
```

El heurístico anteriormente descrito entrega como resultado un conjunto de rutas factibles en términos de la capacidad. En el caso del VRPPC con flota homogénea, este conjunto de rutas puede ser de mayor tamaño que el tamaño de la flota. En particular las instancias de prueba disponibles en la literatura para VRPPC están diseñadas para que no sea posible atender la totalidad de los clientes

con la flota privada. Para modificar la solución devuelta por el heurístico C&W y convertirla en una solución factible del VRPPC con flota homogénea, se propone el siguiente heurístico modificado.

Heurístico C&W modificado

Si $|R| - m$ es mayor que cero, el heurístico C&W modificado elimina $|R| - m$ rutas y asigna los clientes de las rutas eliminadas a la flota pública. Para seleccionar las rutas a eliminar se calcula $\Delta_j = g(r_j) - h(r_j)$ para cada ruta y se eliminan aquellas con mayores valores de Δ_j. El seudocódigo del heurístico C&W modificado (CWM) que se describe a continuación devuelve en R un conjunto de |R| rutas factibles tal que $|R| \leq m$. Los clientes no asignados a alguna de las rutas en R son asignados a la flota pública.

```
R_g ← CWM(G, A, Q, ε, m)
   //Inicialización
   R_g ← CW(G, A, Q, ε)
   for i = 1 to |R_g|
      Δ_i ← g(r_i) - h(r_i)
   next i
   for i = 1 to |R_g| - m
      i* ← argmax{Δ_i}
      Δ_i* ← -∞
      R_g ← R_g \ {r_i*}
   next i
```

4.3 Vecindarios

En este trabajo se proponen dos vecindarios básicos: intercambios e inserciones. En el vecindario de intercambios se seleccionan dos clientes y se intercambian sus posiciones. En el vecindario de inserción se selecciona un cliente, se extrae de su posición actual y se inserta en una posición diferente. En el caso de los intercambios sólo se consideraron intercambios de clientes entre rutas asignadas a la flota privada, mientras que en el caso de las inserciones el cliente puede estar inicialmente asignado tanto a una ruta de la flota privada como a la flota pública. A continuación se describen los vecindarios utilizados en este proyecto.

Intercambio de clientes

El vecindario de intercambio consiste en el conjunto de soluciones diferentes que pueden obtenerse al intercambiar las posiciones de dos vértices asignados a la flota privada. En este proyecto se evaluaron dos variantes para el vecindario de intercambio de clientes: en la primera variante solo se consideran parejas de vértices adyacentes dentro de una misma ruta, mientras que en la segunda variante se consideran vértices asignados a rutas diferentes. Para representar estas dos variantes se definen dos funciones (i.e. V_{INT_A} y V_{INT_B}) que se describen a continuación. Sea $r_s' \leftarrow V_{INT\_A}(r_s, j)$ una función que devuelve en r_s' la ruta resultante luego de intercambiar las posiciones de los vértices j y j+1 en la ruta r_s, tal que $r_s \in R$ y $2 \leq j \leq |r_s| - 2$. En la Figura 4.2 se presenta un ejemplo en una instancia de nueve clientes con una solución arbitraria $S = (R, C)$, tal que $R = \{r_1, r_2\}$ y $C = \{3,8\}$, donde $r_1 = \{0,1,2,5,0\}$ y $r_2 = \{0,7,6,9,4,0\}$. En este caso la función $r_2' \leftarrow V_{INT\_A}(r_2, 3)$ devuelve la ruta $r_2' = \{0,7,9,6,4,0\}$; es decir, devuelve la ruta luego de intercambiar las posiciones del tercero y el cuarto vértice en la ruta original r_2.

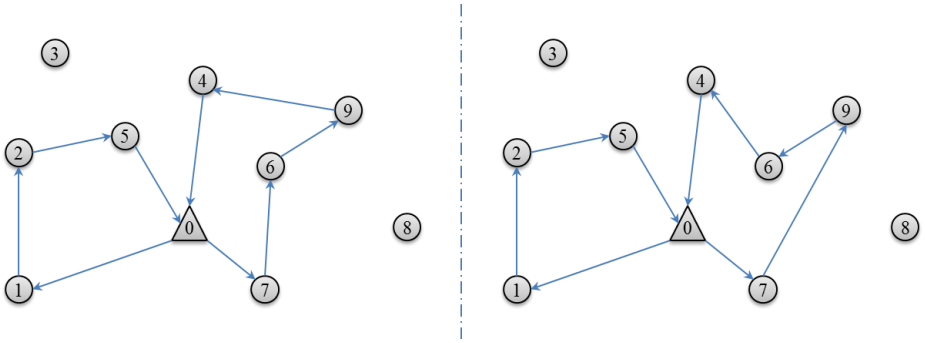


Figura 4.2. Ejemplo de intercambio de vértices adyacentes

En el seudocódigo que se presenta a continuación se describe la forma como se explora la primera variante del vecindario de intercambio de clientes bajo la estrategia BI de búsqueda local (i.e. *best improve*). La función INT_A devuelve en el conjunto de rutas R* un óptimo local luego de explorar el vecindario descrito.

```
R* ← INT_A(R, G, A)
   do
      R* ← R, R_0 ← R
      for s = 1 to |R|
         for j = 2 to |r_s| - 2
            r_s' ← V_INT_A(r_s,j)
            if g(r_s') < g(r_s) then
               R_0 ← R_0\{r_s}, R_0 ← R_0 ∪ {r_s'}
            end if
         next j
      next s
      ΔG ← G(R*) - G(R_0), R* ← R_0
   repeat while ΔG > 0
```

De forma similar se define la función $\{r_s', r_t'\} \leftarrow V_{INT\_B}(r_s, r_t, j, k)$, la cual devuelve en r_s' y r_t' las rutas resultantes luego de intercambiar las posiciones del j-ésimo vértice en r_s y el k-ésimo vértice en r_t, tal que $\{r_s, r_t\} \subseteq R$, $2 \leq j \leq |r_s| - 1$ y $2 \leq k \leq |r_t| - 1$. En la Figura 4.3 se presenta un ejemplo de esta variante de intercambio en una instancia de nueve clientes con una solución arbitraria $S = \{R, C\}$, tal que $R = \{r_1, r_2\}$ y $C = \{3,8\}$, donde $r_1 = \{0,1,2,5,0\}$ y $r_2 = \{0,7,6,9,4,0\}$. En este caso la función $\{r_1', r_2'\} \leftarrow V_{INT\_B}(r_1, r_2, 4,5)$ devuelve las rutas $r_1' = \{0,1,2,4,0\}$ y $r_2' = \{0,7,6,9,5,0\}$; es decir, devuelve las rutas r_1' y r_2' luego de intercambiar el cuarto vértice en r_1 con el quinto vértice en r_2.

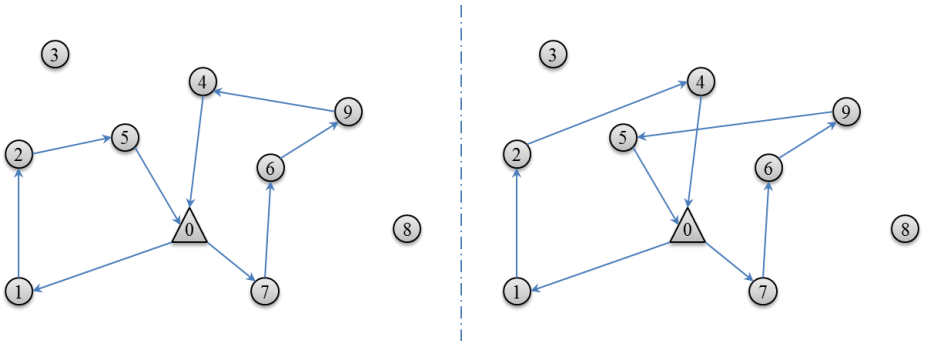


Figura 4.3. Ejemplo de intercambio entre vértices de rutas diferentes

En el seudocódigo que se presenta a continuación se describe la exploración de la segunda variante del vecindario de intercambio de clientes bajo la estrategia BI de búsqueda local. La función INT_B devuelve en el conjunto de rutas R* un óptimo local luego de explorar el vecindario descrito.

```
R* ← INT_B(R, G, A, Q)
   do
      R* ← R, R_0 ← R
      for s = 1 to |R| - 1
         for t = s + 1 to |R|
            for j = 2 to |r_s| - 1
               for k = 2 to |r_t| - 1
                  {r_s', r_t'} ← V_INT_B(r_s,r_t,j,k)
                  if g(r_s') + g(r_t') < g(r_s) + g(r_t) then
                      R_0 ← R_0\{r_s, r_t}, R_0 ← R_0 U {r_s', r_t'}
                  end if
               next k
            next j
         next t
      next s
      ΔG ← G(R*) - G(R_0), R* ← R_0
   repeat while ΔG > 0
```

Inserción de clientes

El vecindario de inserción consiste en el conjunto de soluciones diferentes que pueden obtenerse al extraer un cliente de su posición actual e insertarlo en una posición diferente. Existen tres variantes para el vecindario de inserciones dependiendo del origen y destino de la inserción así: (*i*) de una ruta en la flota privada a una ruta en la flota privada, (*ii*) de ruta en flota privada a flota pública y (*iii*) de flota pública a ruta en flota privada. Para representar estas tres variantes se definen tres funciones (i.e. V_{INS_A}, V_{INS_B} y V_{INS_C}) que se describen a continuación. La función $\{r_s', r_t'\} \leftarrow V_{INS\_A}(r_s, r_t, j, k)$ devuelve en r_s' y r_t' las rutas resultantes luego de extraer el *j*-ésimo vértice en r_s e insertarlo después del *k*-ésimo vértice en r_t, tal que $\{r_s, r_t\} \subseteq R$, $2 \leq j \leq |r_s| - 1$ y $1 \leq k \leq |r_t| - 1$. La Figura 4.4 presenta un ejemplo de la primera variante del vecindario de inserción en una instancia de nueve clientes con una solución arbitraria $S = (R, C)$, tal que $R = \{r_1, r_2\}$ y $C = \{3,8\}$, donde $r_1 = \{0,1,2,5,0\}$ y $r_2 = \{0,7,6,9,4,0\}$. En este caso la función $R' \leftarrow V_{INS\_A}(r_1, r_2, 4,5)$ devuelve el conjunto de rutas $R' = \{r_1', r_2'\}$, donde $r_1' = \{0,1,2,0\}$ y $r_2' = \{0,7,6,9,4,5,0\}$; es decir, devuelve el conjunto de rutas luego de insertar el cuarto vértice en r_1 después del quinto vértice en r_2.

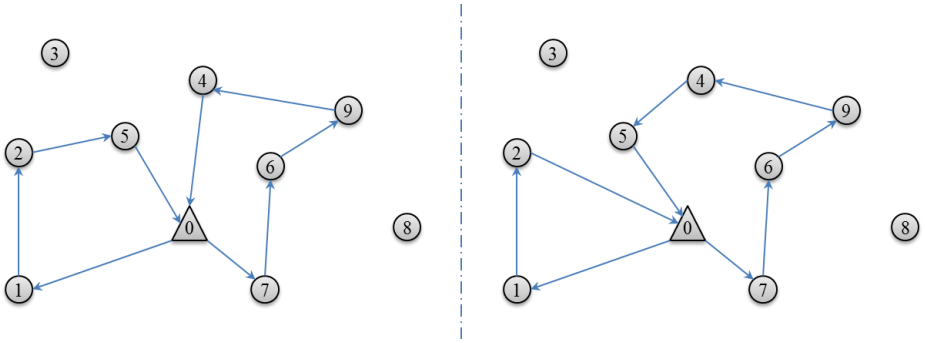


Figura 4.4. Ejemplo de inserción entre rutas de la flota privada

En el seudocódigo que se presenta a continuación se describe la exploración de la primera variante del vecindario de inserción de clientes (i.e. V_{INS_A}) bajo la estrategia BI de búsqueda local. La función INS_A devuelve en el conjunto de rutas R* un óptimo local luego de explorar el vecindario descrito.

```
R* ← INS_A(R, G, A, Q)
   do
      R* ← R, R_0 ← R
      for s = 1 to |R| - 1
         for t = s + 1 to |R|
            for j = 2 to |r_s| - 1
               for k = 1 to |r_t| - 1
                  {r_s', r_t'} ← V_INS_A(r_s,r_t,j,k)
                  if g(r_s') + g(r_t') < g(r_s) + g(r_t) then
                      R_0 ← R_0\{r_s, r_t}, R_0 ← R_0 ∪ {r_s', r_t'}
                  end if
               next k
            next j
         next t
      next s
      ΔG ← G(R*) - G(R_0), R* ← R_0
   repeat while ΔG > 0
```

De manera similar se define la función $\{r_s', C'\} \leftarrow V_{INS\_B}(r_s, j)$ como una función que devuelve en r_s' la ruta resultante luego de extraer el j-ésimo vértice en r_s, y en C' el conjunto de clientes a ser atendidos por la flota pública luego de agregar al conjunto original C el j-ésimo vértice en r_s. En la Figura 4.5 se presenta un ejemplo de la segunda variante del vecindario de inserción en una instancia de nueve clientes con una solución arbitraria $S = \{R, C\}$, tal que $R = \{r_1, r_2\}$ y $C = \{3,8\}$, donde $r_1 = \{0,1,2,5,0\}$ y $r_2 = \{0,7,6,9,4,0\}$. En este caso la función $\{r_s', C'\} \leftarrow V_{INS\_B}(r_1, 4)$ devuelve $r_1' = \{0,1,2,0\}$ y $C' = \{3,5,8\}$; es decir, devuelve el conjunto de rutas luego de asignar el cuarto vértice en r_1 a la flota pública.

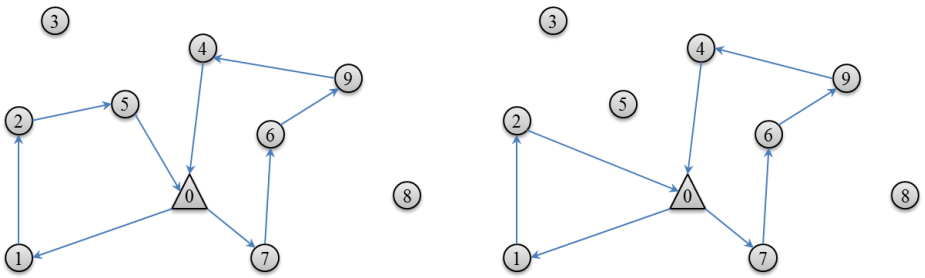


Figura 4.5. Ejemplo de inserción de un cliente de la ruta privada en la flota pública

En el siguiente seudocódigo se describe la forma como se explora la segunda variante del vecindario de inserción de clientes (i.e. V_{INS_B}) bajo la estrategia BI de búsqueda local. La función INS_B devuelve en los conjuntos R* y C* un óptimo local luego de explorar el vecindario descrito.

```
{R*, C*} ← INS_B(R, C, G, A, Q)
   do
      R* ← R, R_0 ← R, C* ← C, C_0 ← C
      for s = 1 to |R|
         for j = 2 to |r_s| - 1
            {r_s', C'} ← V_INS_B(r_s,j)
            if g(r_s') + H(C') < g(r_s) + H(C_0) then
               R_0 ← R_0\{r_s}, R_0 ← R_0 ∪ {r_s'}
               C_0 ← C_0 ∪ {j}
            end if
         next j
      next s
      ΔZ ← G(R*) + H(C*) - G(R_0) - H(C_0), S* ← {R_0, C_0}
   repeat while ΔZ > 0
```

Finalmente, sea $\{r_s', C'\} \leftarrow V_{INS\_C}(r_s, j, v_k)$ una función que devuelve en r_s' la ruta resultante luego de extraer el vértice v_k asignado a la flota pública (i.e. $v_k \in C$) e insertarlo después del j-ésimo vértice en r_s, donde $r_s \in R$ y $2 \leq j \leq |r_s| - 1$. En la Figura 4.6 se presenta un ejemplo de la tercera variante del vecindario de inserción en una instancia con una solución arbitraria inicial $S = (R, C)$, tal que $R = \{r_1, r_2\}$ y $C = \{3,8\}$, donde $r_1 = \{0,1,2,5,0\}$ y $r_2 = \{0,7,6,9,4,0\}$. En este caso la función $\{r_1', C'\} \leftarrow V_{INS\_C}(r_1, 3,1)$ devuelve $r_1' = \{0,1,2,3,5,0\}$ $y\ C' = \{8\}$; es decir, devuelve el conjunto de rutas luego de insertar el vértice 3 asignado inicialmente a la flota pública después del tercer vértice en r_1.

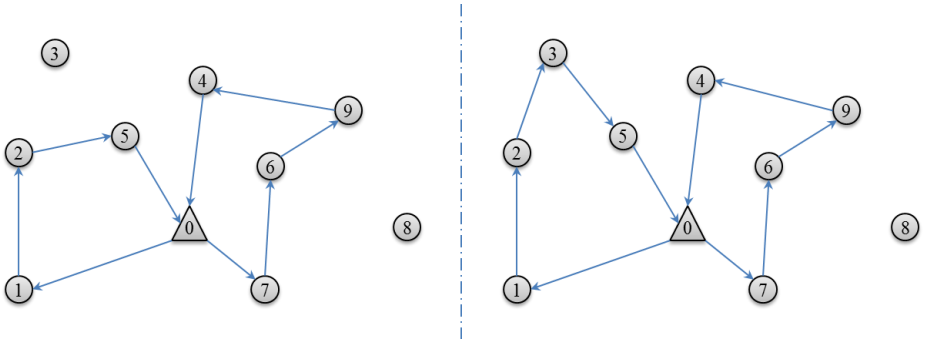


Figura 4.6. Ejemplo de inserción de un cliente de la flota pública en una ruta de la flota privada

En el siguiente seudocódigo se describe la forma como se explora la tercera variante del vecindario de inserción de clientes (i.e. V_{INS_C}) bajo la estrategia BI de búsqueda local. La función INS_C devuelve en los conjuntos R* y C* un óptimo local luego de explorar el vecindario descrito.

```
{R*, C*} ← INS_C(R, C, G, A, Q)
   do
      R* ← R, R_0 ← R, C* ← C, C_0 ← C
      for s = 1 to |R|
         for j = 2 to |r_s| - 1
            for k = 1 to |C|
               {r_s', C'} ← V_INS_C(r_s,j,v_k)
               if g(r_s') + H(C') < g(r_s) + H(C_0) then
                  R_0 ← R_0\{r_s}, R_0 ← R_0 ∪ {r_s'}
                  C_0 ← C_0\{v_k}
               end if
            next k
         next j
      next s
      ΔZ ← G(R*) + H(C*) - G(R_0) - H(C_0), S* ← {R_0, C_0}
   repeat while ΔZ > 0
```

4.4 Heurístico propuesto

El heurístico VNS propuesto toma como datos de entrada una instancia arbitraria Ψ del VRPPC y el conjunto Π de los parámetros del heurístico. La instancia Ψ=[*G*, *M*, *C*] está definida por el grafo G, el conjunto M de los vehículos que componen la flota privada, y la matriz C de distancias. Por su parte el conjunto de parámetros Π=[ε, N] está conformado por los escalares ε y *N*, donde ε es el parámetro con el cual se aleatoriza el cálculo de los ahorros en el heurístico *CW* y *N* es el número de veces que se ejecuta el algoritmo de manera independiente con el fin de diversificar el proceso de búsqueda. La función $\{R^+, C^+\} \leftarrow VNS(\Psi, \Pi)$ devuelve en R^+ un conjunto de rutas factibles a ser ejecutadas por la flota privada y el conjunto C^+ de clientes que van a ser atendidos por la flota pública. A continuación se presenta el seudocódigo del heurístico propuesto y en la Figura 4.7 un diagrama de flujo simple donde se sintetiza el esquema general de funcionamiento del mismo.

```
{R+, C+} ← VNS(Ψ, Π)
for k= 1 to N
   //Solución inicial
   R ← CWM(G, A, Q, ε, m), R+ ← R
   C ← V\R, C+ ← C
   //Intercambio de vértices adyacentes en cada ruta
   R ← INT_A(R, G, A)
   R0 ← R, C0 ← C
   do
      //Intercambio de clientes entre rutas privadas
      do
         R* ← INT_B(R, G, A, Q)
         R* ← INT_A(R*, G, A)
         ΔZ ← G(R) – G(R*), R ← R*
      repeat while ΔZ > 0
      //Inserciones entre clientes de las rutas privadas
      do
         R* ← INS_A(R, G, A, Q)
         R* ← INT_A(R*, G, A)
         ΔZ ← G(R) – G(R*), R ← R*
      repeat while ΔZ > 0
      //Inserción de cliente de ruta privada en flota pública
      do
         {R*, C*} ← INS_B(R, C, G, A, Q)
         R* ← INT_A(R*, G, A)
         ΔZ ← G(R) + H(C) – G(R*) – H(C*), R ← R*, C ← C*
      repeat while ΔZ > 0
      //Inserción de cliente de flota pública en ruta de flota privada
      do
         {R*, C*} ← INS_C(R, C, G, A, Q)
         R* ← INT_A(R*, G, A)
         ΔZ ← G(R) + H(C) – G(R*) – H(C*), R ← R*, C ← C*
      repeat while ΔZ > 0
      Improve ← G(R0) + H(C0) – G(R*) – H(C*), R0 ← R*, C0 ← C*
   repeat while Improve > 0
   //Actualizar mejor solución
   if G(R0) + H(C0) < G(R+) + H(C+) then
      R+ ← R0, C+ ← C0
   end if
next k
```

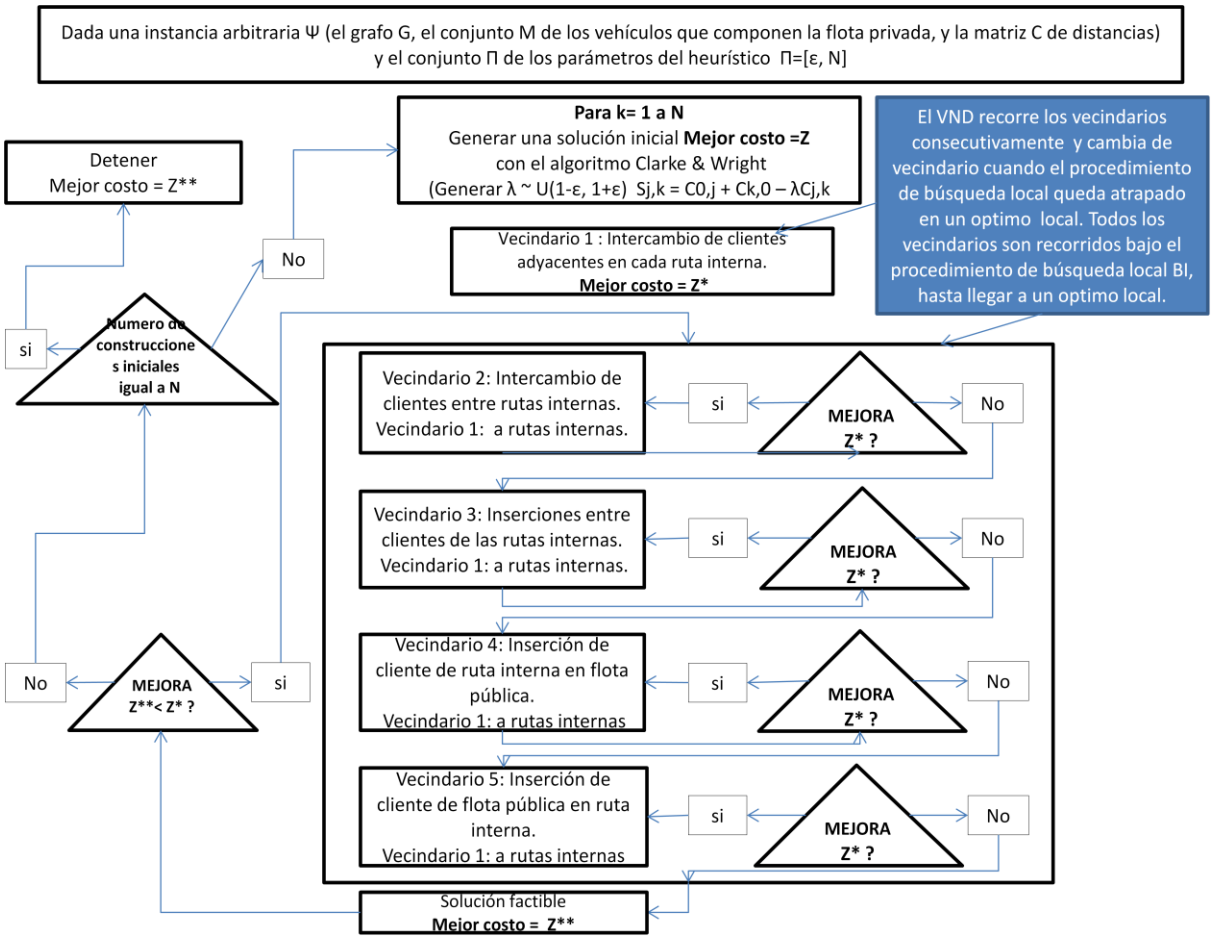


Figura 4.7. Diagrama de flujo de funcionamiento del Heurístico VND propuesto.

5. Resultados Computacionales

En este capítulo se presenta un resumen de los resultados obtenidos a partir de los experimentos computacionales realizados sobre las instancias de prueba descritas en la Tabla 3.2. El heurístico VNS propuesto se implementó en Visual Basic para Aplicaciones (VBA®) y los experimentos computacionales se ejecutaron en el mismo equipo de cómputo utilizado para desarrollar los experimentos descritos en el capítulo 3. En primer lugar se presentan los resultados obtenidos durante la etapa de calibración del heurístico, en la cual se determinan los valores recomendados para los parámetros del heurístico VNS (i.e., ε y N). Una vez determinados los valores de los parámetros se presenta una comparación detallada entre los resultados obtenidos con una réplica del experimento del heurístico propuesto y los mejores resultados de cada método, reportados en la literatura científica sobre el mismo conjunto de instancias de prueba.

5.1 Calibración del heurístico VNS

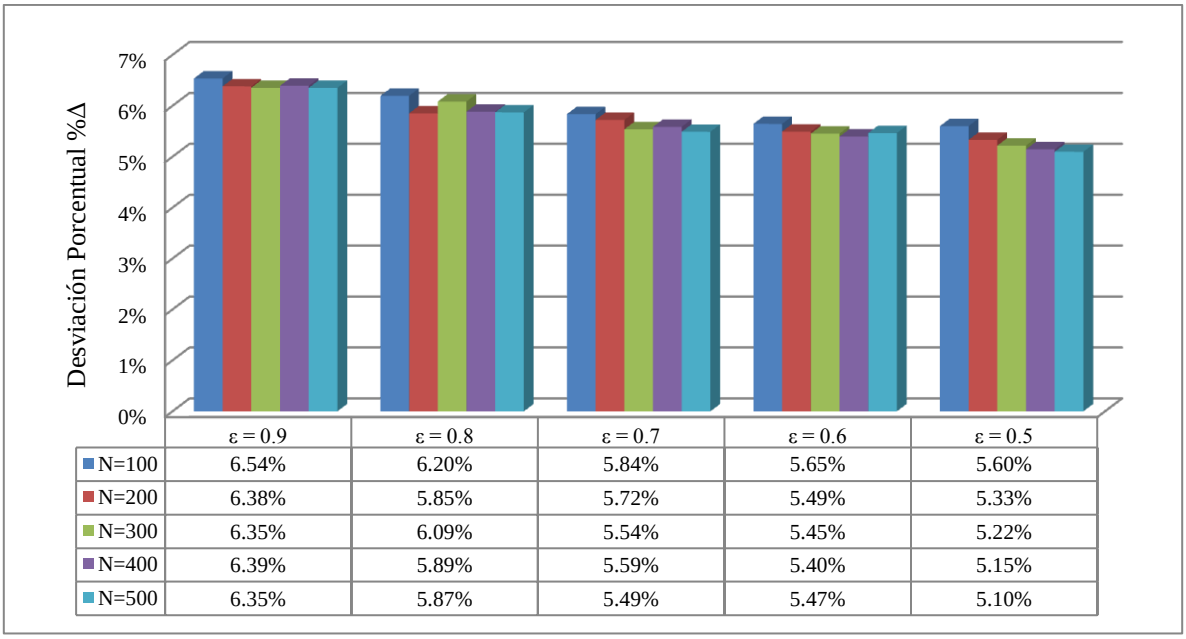
Para determinar los valores de los parámetros del heurístico VNS se llevó a cabo un experimento computacional sobre las 34 instancias de prueba homogéneas. La Tabla 5.1 muestra los valores evaluados para cada uno de los parámetros del heurístico. Cada una de las instancias se resolvió con todas las posibles combinaciones de estos dos parámetros, y en cada caso se calculó la diferencia porcentual entre la solución obtenida por el heurístico VNS (i.e. Z_{VNS}) y el valor de referencia R (i.e. Z_R), calculado como en la ecuación 5.1. Para el análisis se tomaron tres valores de referencia: (i) la solución del heurístico RIP (i.e. Z_{RIP}), (ii) la solución reportada por el algoritmo genético (i.e. Z_{GA}), y (iii) la mejor solución encontrada en la literatura (Z*).

Tabla 5.1. Valores evaluados para los parámetros del heurístico

Parámetro	*Valores*
ε	{0.1, 0.2, 0.3, 0.4, 0.5}
N	{100, 200, 300, 400, 500}

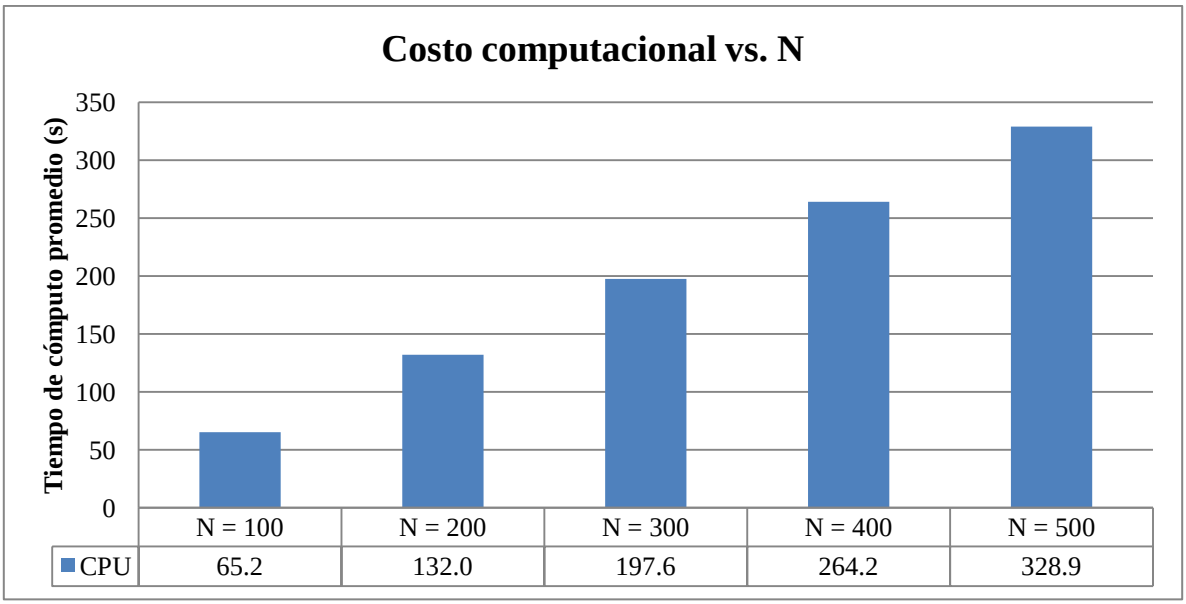
$$\%\Delta(R) = \frac{Z_{VNS} - Z_R}{Z_R} \times 100\% \qquad 5.1$$

En la Figura 5.1 se puede observar como los valores promedio de las diferencias, calculadas como se explicó anteriormente, disminuyen a medida que aumenta el valor del parámetro ε. De este análisis se desprende que el valor del parámetro **ε** que ofrece el mejor desempeño es ε=0.5. Así mismo se puede observar como los valores promedio de las diferencias disminuyen a medida que aumenta el valor del parámetro N. Por lo tanto, el valor del parámetro N que ofrece el mejor desempeño es N=500. Aunque la tendencia observada en la Figura 5.1 hace pensar que los resultados podrían mejorar si se incrementa el número de arranques para el heurístico propuesto (i.e. el valor del parámetro N) en la Figura 5.2 se observa cómo, a medida que aumenta el número de arranques para el heurístico, el costo computacional (i.e. CPU) aumenta considerablemente. Así por ejemplo, el resultado obtenido con N=500 es ligeramente mejor que el obtenido con N=400, pero el primero requiere casi el 20% más tiempo de cómputo.



	ε = 0.9	ε = 0.8	ε = 0.7	ε = 0.6	ε = 0.5
N=100	6.54%	6.20%	5.84%	5.65%	5.60%
N=200	6.38%	5.85%	5.72%	5.49%	5.33%
N=300	6.35%	6.09%	5.54%	5.45%	5.22%
N=400	6.39%	5.89%	5.59%	5.40%	5.15%
N=500	6.35%	5.87%	5.49%	5.47%	5.10%

Figura 5.1. Resultados computacionales para cada combinación de los parámetros



	N = 100	N = 200	N = 300	N = 400	N = 500
CPU	65.2	132.0	197.6	264.2	328.9

Figura 5.2. Análisis del tiempo computacional respecto al parámetro N

Del análisis de los resultados obtenidos para las 34 instancias homogéneas del VRPPC, la combinación de parámetros ε=0.5 y N= 500 fue seleccionada por ofrecer un compromiso aceptable entre calidad de la solución y costo computacional.

5.2 Desempeño del heurístico VNS

Para evaluar el desempeño del heurístico VNS propuesto se resolvieron las 34 instancias de prueba con flota homogénea descritas en la Tabla 3.2. Los resultados obtenidos para cada instancia se presentan en la Tabla 5.2: la columnas en la tabla presentan respectivamente el nombre de la

instancia, el costo total obtenido (i.e. Z_{VNS}) y el tiempo de cómputo requerido por el heurístico VNS para terminar (i.e. CPU) en segundos.

Tabla 5.2. Resultados obtenidos por el heurístico VNS

Instancia	Z_{vns}	*CPU (s)*
CE-01.txt	1178.63	5.31
CE-02.txt	1864.64	13.54
CE-03.txt	2067.65	21.21
CE-04.txt	2643.43	48.11
CE-05.txt	3218.26	89.00
CE-06.txt	1268.63	5.13
CE-07.txt	2060.64	13.24
CE-08.txt	2206.65	20.31
CE-09.txt	2479.43	52.47
CE-10.txt	3506.43	91.86
CE-11.txt	2381.12	30.54
CE-12.txt	2036.43	26.69
CE-13.txt	2913.12	30.93
CE-14.txt	2364.94	19.46

Instancia	Z_{vns}	*CPU*
G-01.txt	14552.99	162.62
G-02.txt	19824.43	351.40
G-03.txt	26192.98	606.39
G-04.txt	36944.66	828.80
G-05.txt	15319.93	127.53
G-06.txt	23774.23	252.02
G-07.txt	25143.95	533.93
G-08.txt	32064.8	622.17
G-09.txt	1362.412	289.41
G-10.txt	1633.527	529.50
G-11.txt	2217.398	821.15
G-12.txt	2559.544	1308.41
G-13.txt	2340.728	249.22
G-14.txt	2804.285	481.55
G-15.txt	3242.292	842.19
G-16.txt	3735.34	1495.76
G-17.txt	1748.433	152.34
G-18.txt	2918.461	251.55
G-19.txt	3779.463	356.89
G-20.txt	4628.694	546.02

Adicionalmente, para cada instancia se calculó el porcentaje de desviación entre la solución obtenida por el heurístico VNS (i.e. Z_{VNS}) y el valor de referencia R (i.e. Z_R), calculado como en la ecuación 5.1. Para el análisis se tomaron como valores de referencia los resultados reportados en la literatura para las mismas instancias: (i) la solución del heurístico RIP (i.e. Z_{RIP}) propuesto en [45], (ii) la solución reportada por el algoritmo genético (i.e. Z_{GA}) propuesto en [42], (iii) la solución reportada por el algoritmo de búsqueda tabú (i.e. Z_{TS}) propuesto en [17], (iv) la solución reportada por el algoritmo de búsqueda tabú (i.e. Z_{TS+}) propuesto en [46], y (v) la solución reportada por el algoritmo AVNS (i.e. Z_{AVNS}) propuesto en [21]. En la Figura 5.3 se presenta un diagrama de caja o *box-plot* a manera de resumen de los resultados obtenidos.

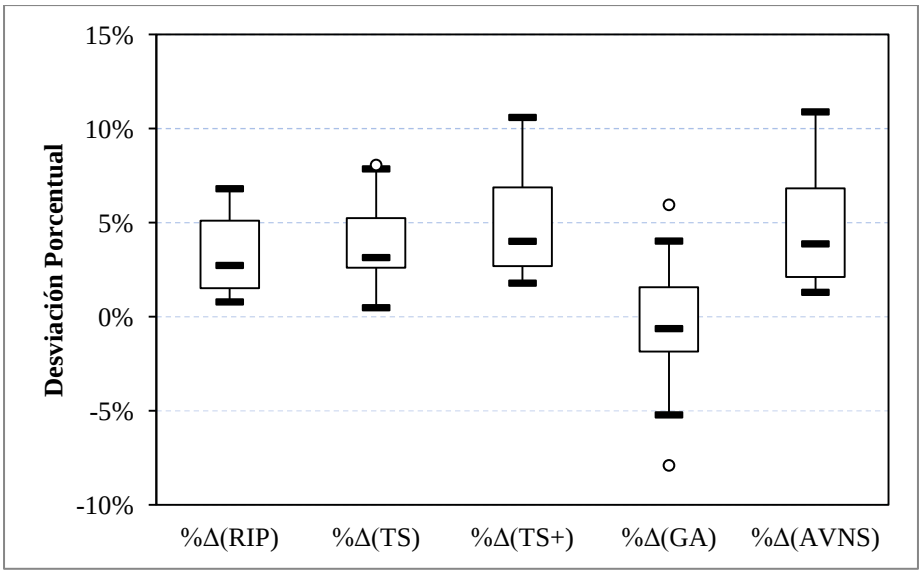


Figura 5.3. Desviación porcentual promedio del heurístico VNS respecto a los resultados de la literatura

En la Figura 5.4 se presenta un gráfico similar para comparar el costo computacional de los métodos evaluados, incluido el heurístico VNS propuesto. Aunque la comparación de tiempos de ejecución no incluye el equipo de cómputo utilizado en cada caso, la gráfica permite apreciar cómo el heurístico propuesto requiere un tiempo de cómputo relativamente bajo y presenta poca varianza en comparación con los otros métodos evaluados.

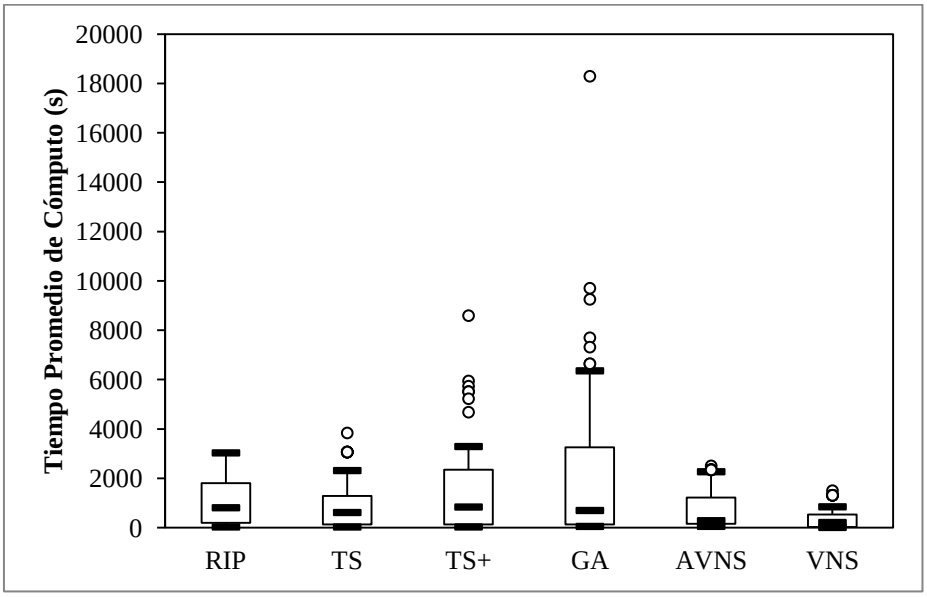


Figura 5.4. Comparación de los tiempos computacionales de los métodos evaluados

Por último, en la Tabla 5.3 se presenta un reporte detallado con los resultados de cada uno de los métodos de solución que fueron usados como referencia para cada una de las instancias de prueba usadas en el experimento y se incluye la desviación del método propuesto con respecto a cada método.

Tabla 5.3. Comparación de resultados

Instancia	*RIP*			*TS*			*TS+*			*GA*			*AVNS*		
	Z_{RIP}	*CPU(seg)*	$\Delta\%(Z_{RIP})$	Z_{TS}	*CPU(seg)*	$\Delta\%(Z_{TS})$	Z_{TS+}	*CPU(seg)*	$\Delta\%(Z_{TS+})$	Z_{GA}	*CPU(seg)*	$\Delta\%(Z_{GA})$	Z_{AVNS}	*CPU(seg)*	$\Delta\%(Z_{AVNS})$
CE-01.txt	1132.91	25	4.04%	1119.47	24.3	5.28%	1119.47	24.9	5.28%	1158.98	48.79	1.70%	1123.95	92.5	4.87%
CE-02.txt	1835.76	73	1.57%	1814.52	33	2.76%	1814.52	33.9	2.76%	1893.66	90.21	-1.53%	1814.52	48.6	2.76%
CE-03.txt	1959.65	107	5.51%	1921.1	78.6	7.63%	1930.66	81	7.10%	1987.75	104.08	4.02%	1920.86	212.1	7.64%
CE-04.txt	2545.72	250	3.84%	2525.17	193.2	4.68%	2525.17	200.6	4.68%	2668.87	191.66	-0.95%	2512.05	279.7	5.23%
CE-05.txt	3172.22	474	1.45%	3113.58	309.9	3.36%	3117.1	353.2	3.25%	3279.64	308.91	-1.87%	3099.77	228.6	3.82%
CE-06.txt	1208.33	25	4.99%	1207.47	25.5	5.07%	1207.47	25.2	5.07%	1233.2	46.6	2.87%	1207.81	75.9	5.04%
CE-07.txt	2006.52	71	2.70%	2006.52	32.7	2.70%	2006.52	34	2.70%	2086.17	91.1	-1.22%	2013.93	50.9	2.32%
CE-08.txt	2082.75	110	5.95%	2060.17	85.1	7.11%	2056.59	81.6	7.30%	2130.82	107.77	3.56%	2052.05	253.1	7.53%
CE-09.txt	2443.94	260	1.45%	2438.43	185.3	1.68%	2435.97	188.2	1.78%	2558.7	204.95	-3.10%	2432.51	259	1.93%
CE-10.txt	3464.9	478	1.20%	3406.82	311.1	2.92%	3401.83	345.7	3.07%	3598.36	314.97	-2.55%	3391.35	201	3.39%
CE-11.txt	2333.03	195	2.06%	2353.39	126.3	1.18%	2332.36	131	2.09%	2383.34	125.61	-0.09%	2332.21	316	2.10%
CE-12.txt	1953.55	128	4.24%	1952.86	60.4	4.28%	1952.86	59.5	4.28%	2042.84	105.74	-0.31%	1953.55	92.9	4.24%
CE-13.txt	2864.21	188	1.71%	2882.7	130	1.06%	2860.89	132.1	1.83%	2929.02	125.08	-0.54%	2858.94	278.5	1.90%
CE-14.txt	2224.63	110	6.31%	2216.97	65	6.67%	2216.97	64.2	6.67%	2338.22	101.85	1.14%	2215.38	93.2	6.75%
G-01.txt	14388.58	651	1.14%	14218.83	638.3	2.35%	14190.01	1183.8	2.56%	14910.52	479.62	-2.40%	14157.08	652.6	2.80%
G-02.txt	19505	1178	1.64%	19729.96	1215.2	0.48%	19208.52	5220.6	3.21%	20258.91	1699.07	-2.14%	19204.36	1558.4	3.23%
G-03.txt	24978.17	2061	4.86%	25653.58	2241.3	2.10%	24592.18	5940.4	6.51%	25941.17	7691.85	0.97%	24602.61	2356.1	6.46%
G-04.txt	34957.98	3027	5.68%	36022.73	3833.9	2.56%	34802.08	5508.7	6.16%	36083.77	9697.44	2.39%	34415.82	2500.9	7.35%
G-05.txt	14683.03	589	4.34%	14673.56	875	4.41%	14261.31	847.8	7.42%	14875.44	1002.59	2.99%	14272.32	1301.1	7.34%
G-06.txt	22260.19	1021	6.80%	22278.99	1445.3	6.71%	21498.03	1591.1	10.59%	22440.03	3231.16	5.95%	21440.79	1783.5	10.88%
G-07.txt	23963.36	1628	4.93%	24191.41	2052.8	3.94%	23513.06	5514	6.94%	24621.42	6638.28	2.12%	23375.6	2262.8	7.56%
G-08.txt	30496.18	2419	5.14%	30627.91	3059.9	4.69%	30073.56	5729	6.62%	31326.38	7311.45	2.36%	29797.62	2339.7	7.61%
G-09.txt	1341.17	832	1.58%	1328.14	611	2.58%	1325.62	819.1	2.78%	1368.47	2032.36	-0.44%	1335.45	602	2.02%
G-10.txt	1612.09	1294	1.33%	1590.83	938.8	2.68%	1590.82	1762.3	2.68%	1646.2	5633.33	-0.77%	1604.5	978.4	1.81%
G-11.txt	2198.45	2004	0.86%	2172.28	1492.7	2.08%	2173.8	3284.3	2.01%	2235.24	9246.96	-0.80%	2189.02	1534.3	1.30%
G-12.txt	2521.79	2900	1.50%	2492.75	2309.7	2.68%	2495.02	8587.6	2.59%	2578.12	18287.61	-0.72%	2520.29	2043.9	1.56%
G-13.txt	2286.91	802	2.35%	2278.99	360.8	2.71%	2274.12	504.5	2.93%	2347.49	772.48	-0.29%	2291.83	116.5	2.13%
G-14.txt	2750.75	1251	1.95%	2705	610.4	3.67%	2703.31	976.9	3.74%	2796.74	1487.42	0.27%	2708.22	183.5	3.55%
G-15.txt	3216.99	1862	0.79%	3158.92	924.8	2.64%	3161.26	1952	2.56%	3283.07	3264.51	-1.24%	3194.82	357.3	1.49%
G-16.txt	3693.62	2778	1.13%	3639.11	1313.7	2.64%	3638.39	4675.1	2.66%	3804.04	6351.35	-1.81%	3671.34	561.2	1.74%
G-17.txt	1701.58	806	2.75%	1636.11	402.6	6.87%	1633.35	472	7.05%	1898.36	329.98	-7.90%	1682.49	110.3	3.92%
G-18.txt	2765.92	1303	5.52%	2705.9	622	7.86%	2710.21	892	7.68%	3079.03	613.52	-5.21%	2741.8	156.4	6.44%
G-19.txt	3576.92	1903	5.66%	3497.54	1012.5	8.06%	3497.72	1418.4	8.06%	3940.71	878.97	-4.09%	3507.94	194.1	7.74%
G-20.txt	4378.13	2800	5.72%	4311.17	1368.9	7.37%	4306.89	2476.3	7.47%	4823.76	1085.47	-4.04%	4332.44	290.3	6.84%
	Promedio		**3.31%**	**Promedio**		**3.98%**	**Promedio**		**4.74%**	**Promedio**		**-0.40%**	**Promedio**		**4.51%**

6. Conclusiones

El problema de la planeación de rutas de vehículos en sistemas en los que se combinan una flota de vehículos propia o privada y un operador público con quien se pueden subcontratar algunas de las visitas a un conjunto de clientes es un problema computacionalmente complejo que ha recibido relativamente poca atención en la literatura científica. Experimentos computacionales desarrollados sobre un conjunto de instancias de prueba disponibles para el problema permitieron demostrar empíricamente que solo es posible resolver por métodos exactos, (i.e. *Branch and Bound*), utilizando su formulación como un programa entero mixto en software de optimización comercial, instancias relativamente pequeñas con hasta diez clientes y dos vehículos en una hora de cómputo o menos. Para instancias de mayor tamaño obtener soluciones exactas se hace inviable o excesivamente costoso computacionalmente. Para resolver instancias para las cuales los métodos exactos no son una alternativa viable, en este proyecto se propone un nuevo método heurístico de solución basado en el paradigma de la búsqueda en vecindario variable, más conocido en la literatura como VNS. Los resultados computacionales obtenidos sobre el mismo conjunto de instancias usado para probar los métodos exactos permiten concluir que el heurístico propuesto es efectivo en resolver instancias de mayor tamaño. Las soluciones obtenidas son comparables en términos de calidad de la solución y tiempo computacional respecto al desempeño de los otros métodos publicados para el mismo problema. Particularmente, los experimentos computacionales muestran que el heurístico propuesto obtiene soluciones con una desviación porcentual respecto a los métodos de solución disponibles en la literatura que oscila entre -7.9% y 10.88%. El costo computacional del heurístico propuesto, también de acuerdo con los experimentos computacionales desarrollados, es razonable y presenta poca varianza.

Referencias

[1] R. Engelman y Worldwatch Institute, "Estado de la población mundial," 2009.

[2] R. C. Larson y A. R. Odoni, *Urban Operations Research*. Prentice-Hall, NJ, 1981, p. 573.

[3] S. Chopra y P. Meindl, *Administración de la cadena de suministro: estrategia, planeación y operación*. Pearson Educacion, 2008, p. 536.

[4] U.S Bureau of Transportation Statics, "Freight Transportation: Global Highlights 2010," 2010.

[5] C. A. Peñaloza Pabon, "Diagnostico del transporte 2009," Bogotá, Colombia, 2009.

[6] A. U. Gallego Henao, "Estructura de costos de operación vehicular 2006," Bogotá, Colombia, 2007.

[7] C. Sutcliffe y J. Boardman, "The Ex-Ante Benefits of Solving Vehicle-Routing Problems," *The Journal of the Operational Research Society*, vol. 42, no. 2, pp. 135–143, 1991.

[8] P. Toth y D. Vigo, *The Vehicle Routing Problem*. Bolonia, Italia: siam Monographs and discrete mathematics and applications, 2002, p. 386.

[9] C. Haksever, B. Render, R. Russell, and R. Murdick, "Routing Problems and Scheduling," in *Service Management and Operations*, 2nd ed., Prentice Hall, 2000, pp. 476–497.

[10] B. Eksioglu, A. V. Vural, y A. Reisman, "The vehicle routing problem: A taxonomic review," *Computers & Industrial Engineering*, vol. 57, no. 4, pp. 1472–1483, Nov. 2009.

[11] D. O. Bausch, G. G. Brown, y D. Ronen, "Dispatching shipments at minimal cost with multiplie mode aternatives," *Journal of business logistics*, vol. 15, no. 1, pp. 287–304, 1994.

[12] M. Fisher, "Vehicle Routing," in *Network Routing*, Vol 8., 1995, pp. 1–33.

[13] R. Baldacci, E. Hadjiconstantinou, y a. Mingozzi, "An Exact Algorithm for the Capacitated Vehicle Routing Problem Based on a Two-Commodity Network Flow Formulation," *Operations Research*, vol. 52, no. 5, pp. 723–738, Oct. 2004.

[14] P. Kilby y P. Shaw, "Vehicle Routing," in *Handbook of constraint programming*, Elsevier B.V., 2006, p. 977.

[15] J. Euchi y H. Chabchoub, "Heuristic Search Techniques to Solve the Vehicle Routing with Private Fleet y Common Carrier," *International Journal*, 2010.

[16] M.-C. Bolduc, J. Renaud, y F. Boctor, "A heuristic for the routing and carrier selection problem," *European Journal of Operational Research*, vol. 183, no. 2, pp. 926–932, Dec. 2006.

[17] J.-F. Côté y J.-Y. Potvin, "A tabu search heuristic for the vehicle routing problem with private fleet and common carrier," *European Journal of Operational Research*, vol. 198, no. 2, pp. 464–469, Oct. 2009.

[18] J. Euchi y H. Chabchoub, "Iterated Density Estimation with 2-opt local search for the vehicle routing problem with private fleet and common carrier," *2009 International Conference on Computers Industrial Engineering*, pp. 1058–1063, 2009.

[19] H. Kopfer y M. A. Krajewska, "Approaches for Modelling and Solving the Integrated Transportation and Forwarding Problem," *Logistic management*, 2007.

[20] P. Hansen, N. Mladenović, y J. a. Moreno Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, vol. 175, no. 1, pp. 367–407, Oct. 2009.

[21] A. Stenger, D. Vigo, S. Enz, y M. Schwind, "An Adaptive Variable Neighborhood Search Algorithm for a Vehicle Routing Problem Arising in Small Package Shipping," *Transportation Science*, vol. 1655, pp. 1–17, Feb. 2012.

[22] G. B. Dantzig y J. H. Ramser, "The Truck Dispatching Problem," *Management Science*, vol. Vol. 6, pp. 80–91, 1959.

[23] G. Clarke y J. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, vol. 12, no. 4, pp. 586–581, 1964.

[24] B. Chandran y S. Raghavan, "Modeling and Solving the Capacitated Vehicle Routing Problem on Trees," in *The Vehicle Routing problem: Latest advances and new challenges*, vol. 43, B. Golden, S. Raghavan, y E. Wasil, Eds. Boston, MA: Springer US, 2008, pp. 239–262.

[25] M. Gendreau, G. Laporte, C. Musaraganyi, y D. D. Taillard, "A tabu search heuristic for the heterogeneous fleet vehicle routing problem," *Computers & Operations Research*, vol. 26, pp. 1153–1173, 1999.

[26] F. Liu y S. Shen, "The fleet size and mix vehicle routing problem with time windows," *The Journal of the Operational Research Society*, vol. 50, no. 7, pp. 721–732, 2010.

[27] I. . Giosa, O. Viera, y L. Tansini, “New assignment algorithms for the multi-depot vehicle routing problem,” *Journal of the Operational Research Society*, vol. 53, pp. 977–984, 2002.
[28] B. Barán, “Comparación de un Sistema de Colonias de Hormigas y una Estrategia Evolutiva para el Problema del Ruteo de Vehículos con Ventanas de Tiempo en un Contexto Multiobjetivo.”
[29] L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, y L. Paquete, “Metaheuristics for the Vehicle Routing Problem with Stochastic Demands,” *Lecture notes Computer Science*, vol. 3242/2004, no. 1, pp. 450–460, 2004.
[30] R. Baldacci, E. Hadjiconstantinou, y A. Mingozzi, “An exact algorithm for the capacitated vehicle routing problem based on two commodity network flow formulation,” *Operations Research*, vol. 52, no. No. 5, pp. 723–738, 2004.
[31] B. Kallehauge, “Formulations and exact algorithms for the vehicle routing problem with time windows,” *Computers & Operations Research*, vol. 35, no. 7, pp. 2307–2330, 2008.
[32] P. Toth y D. Vigo, “Models, relaxations and exact approaches for the capacitated vehicle routing problem,” *Discrete Applied Mathematics*, vol. 123, no. 1–3, pp. 487–512, Nov. 2002.
[33] J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, y F. Semet, “A guide to vehicle routing heuristics,” *Journal of the Operational Research Society*, vol. 53, no. 5, pp. 512–522, May 2002.
[34] İ. K. Altınel y T. Öncan, “A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem,” *Journal of the Operational Research Society*, vol. 56, pp. 954–961, 2005.
[35] J. Renaud y F. F. Boctor, “A sweep-based algorithm for the fleet size and mix vehicle routing problem,” *European Journal of Operational Research*, vol. 140, no. 3, pp. 618–628, 2002.
[36] I. H. Osman, “Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem,” *Annals of Operations Research*, vol. 41, no. 4, pp. 421–451, 1993.
[37] F. Glover y G. A. Kochenberger, *Handbook of Metaheuristics*, vol. 146. Boston, MA: Springer US, 2010, p. 600.
[38] M. Gendreau, J.-Y. Potvin, O. Bräumlaysy, G. Hasle, y A. Løkketangen, “Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography,” in *The Vehicle Routing problem: Latest advances and new challenges*, 2008, pp. 143–169.
[39] W.-C. Chiang y R. A. Russell, “Simulated annealing metaheuristics for the vehicle routing problem with time windows,” *Annals of Operations Research*, vol. 63, no. 1, pp. 3–27, 1996.
[40] M.-C. Bolduc, G. Laporte, J. Renaud, y F. F. Boctor, “A tabu search heuristic for the split delivery vehicle routing problem with production and demand calendars,” *European Journal of Operational Research*, vol. 202, no. 1, pp. 122–130, Apr. 2010.
[41] S. Ghariani y V. Furnon, “Constraint Programming and Ant Colonies Applied to Vehicle Routing Problems,” in *Metaheuristics for hard optimization*, 2006, pp. 307–326.
[42] J. Kratica, T. Kostic, D. Tosic, D. Dugosija, y V. Filipovic, “A Genetic Algorithm for the Routing and Carrier Selection Problem ⋆,” *Computer Science and Information Systems*, vol. In Press.
[43] M. Diaby y R. Ramesh, “The Distribution Problem with Carrier Service: A Dual Based Penalty Approach,” *ORSA Journal on computing*, vol. 7, pp. 24–35, 1995.
[44] C.-W. Chu, “A heuristic algorithm for the truckload and less-than-truckload problem,” *European Journal of Operational Research*, vol. 165, no. 3, pp. 657–667, Sep. 2005.
[45] M. Bolduc, J. Renaud, F. Boctor, y G. Laporte, “A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers,” *Journal of the Operational Research Society*, pp. 776–787, 2008.
[46] J. Potvin y M.-A. Naud, “Tabu Search with Ejection Chains for the Vehicle Routing Problem with Private Fleet and Common,” *Journal of the Operational Research Society*, vol. 62, pp. 326–336, 2011.
[47] J. Euchi, H. Chabchoub, y A. Yassine, “New Evolutionary Algorithm Based on 2-Opt Local Search to Solve the Vehicle Routing Problem with Private Fleet and Common Carrier,” *International Journal of Applied Metaheuristic Computing*, vol. 2, no. March, pp. 58–82, 2011.
[48] J. G. Klincewicz, H. Luss, y M. G. Pilcher, “Fleet Size Planing When Outside Carrier Services Are Available,” *Transportation Science*, vol. 24, no. 3, pp. 169–182, 1990.
[49] R. W. Hall y M. Racer, “Transportation with common carrier and private fleets: system assignment and shipment frequency optimization,” *IIE Transactions*, vol. 27, no. 2. pp. 217–225, Apr-1995.
[50] N. Christofides y S. Eilon, “An Algorithm for the Vehicle- dispatching Problem,” *Journal of the Operational Research Society*, vol. 20, no. 3, pp. 309–318, 1969.

[51] B. L. Golden, E. A. Wasil, J. P. Kelly, y I.-M. Chao, "The impact of metaheuristics on solving the vehicle routing," in *Fleet management and logistics*, Kluwer, Ed. Boston: , 1998, pp. 33–56.
[52] M.-C. Bolduc, "Instances for the Vehicle Routing Problem with Private fleet and Common Carrier (VRPPC)," 2012. [Online]. Available: http://www.mcbolduc.com/tests.htm.
[53] N. Mladenovic and P. Hansen, "Variable Neighborhood Search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.
[54] P. C. Yellow, "A Computational Modification to the Savings Method of Vehicle Scheduling," *Operational Research Quarterly*, vol. 21, no. 2, pp. 281–283, 1970.
[55] B. L. Golden, T. L. Magnanti, y H. Q. Nguyen, "Implementing vehicle routing algorithms," *Networks*, vol. 7, no. 2, pp. 113–148, 1977.