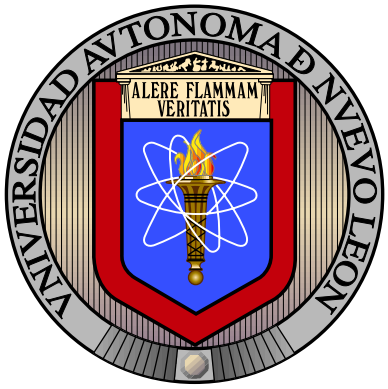
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado



Un enfoque metaheurístico para un problema de ruteo con flexibilidad en las fechas de entrega

por

Irma Delia García Calvillo

en opción al grado de

Doctor en Ingeniería

con especialidad en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León junio 2010

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Un enfoque metaheurístico para un problema de ruteo con flexibilidad en las fechas de entrega

por

Irma Delia García Calvillo

en opción al grado de

Doctor en Ingeniería

con especialidad en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León junio 2010

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Un enfoque metaheurístico para un problema de ruteo con flexibilidad en las fechas de entrega», realizada por la alumna Irma Delia García Calvillo, con número de matrícula 01440304, sea aceptada para su defensa como opción al grado de Doctor en Ingeniería con especialidad en Ingeniería de Sistemas.

El Comité de Tesis

Dra. Ada Margarita Álvarez Socarrás
Asesor

Dr. Joaquín Pacheco Bonrostro
Asesor

Dr. Fernando López Irarragorri
Revisor

Dr. Igor S. Litvinchev
Revisor

Dr. Pablo Barrera Sánchez
Revisor

Vo. Bo.

Dr. Moisés Hinojosa Rivera
División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, junio 2010

A Miguel Ángel por su esfuerzo compartido.

A Sebastián y Diego por compartirme sus risas, juegos y alegrías.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Las mejores enseñanzas son las que se viven y aprenden de personas que logran transmitir no sólo sus conocimientos académicos sino enseñanzas para la vida. He aprendido mucho en el camino, sobre todo enseñanzas de vida, de las que no se olvidan y deseo agradecer a las personas e instituciones que lo hicieron posible.

En primer lugar mi mas sincero agradecimiento a la Dra. Ada Margarita Álvarez Socarrás por ser mi guía en mi formación académica, por su asesoría, dedicación e infinita paciencia para el desarrollo de esta tesis, pero sobre todo por su amistad, su confianza, comprensión y enseñanzas fuera del aula, gracias.

Al Dr. Joaquín Pacheco Bonrostro, le agradezco profundamente haberme guiado en la realización de esta tesis, por compartirme sus conocimientos y experiencia, gracias por su paciencia, dedicación y trabajo conjunto, pero sobre todo gracias por su amistad y hospitalidad.

Gracias a los profesores del Programa de Posgrado de Ingeniería en Sistemas de la Universidad Autónoma de Nuevo León por su compromiso en la formación de recursos humanos de calidad, es un orgullo pertenecer a este Posgrado.

Agradezco de manera especial a los revisores de este trabajo de tesis, al Dr. Fernado López, al Dr. Igor Litvinchev y al Dr. Pablo Barrera, por la lectura detallada de este documento y por los comentarios y sugerencias que fueron de gran relevancia para la redacción del mismo.

Mi agradecimiento a la Universidad Autónoma de Nuevo León por el apoyo económico otorgado para la realización de mis estudios de Doctorado. Asimismo, deseo agradecer el apoyo de la Universidad Autónoma de Coahuila para la realización de mis estancias en la Universidad de Burgos. De manera especial al Director del Centro de Investigación en Matemáticas Aplicadas, M.C. Francisco Javier Cepeda Flores, por la confianza y apoyo incondicional.

Gracias al Departamento de Economía Aplicada de la Universidad de Burgos por la hospitalidad y todas las facilidades brindadas para la realización de las estancias académicas que fueron decisivas para la elaboración de esta tesis. En particular a la Dra. Cristina Delgado y la Dra. Isabel González.

Gracias a mis amigos y compañeros que me alentaron en todo momento, en especial a la M.C. Yajaira Cardona y M.C. Vanesa Avalos por su amistad y constante apoyo.

Al Dr. Humberto Madrid de la Vega gracias por su motivación, confianza, su amistad, por tener siempre un buen consejo.

A mi familia, les agradezco su cariño y aliento, en especial a mis Padres Irma Calvillo y Juvenal García, gracias por los valores, la formación, el ejemplo y la educación que me han brindado, por contar siempre con ustedes.

Por último, a quienes me impulsaron en todo momento, no me dejaron claudicar, gracias por ser mi soporte, mi apoyo y mi razón de continuar, a Miguel Angel y mis hijos Sebastián y Diego; gracias por ser, por existir, por vivir en mí, por acompañarme en esta aventura y en las que están por venir. Gracias a Dios por poder compartir la satisfacción de cumplir una meta mas.

Irma Delia García Calvillo

San Nicolás de los Garza, Nuevo León, junio 2010

Resumen

Irma Delia García Calvillo.

Candidato para el grado de Doctor en Ingeniería
con especialidad en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

Un enfoque metaheurístico para un problema de ruteo con flexibilidad en las fechas de entrega

Número de páginas: 231.

Objetivos y método de estudio: El objetivo del presente trabajo es el estudio de un problema de ruteo de vehículos inspirado en una situación real que afronta una empresa distribuidora de productos alimenticios en España. La empresa desea planificar de forma óptima la entrega de sus productos a sus centros de distribución en un horizonte de planeación dado.

Con el fin de reducir los costos de transportación, se propone permitir ciertas y controladas variaciones en las fechas inicialmente establecidas por los centros de distribución para recibir los productos solicitados, y se considera el problema de ruteo sobre todo el horizonte de planeación.

Al problema que se obtiene al considerar la política de adelanto de pedidos es lo que hemos denominado problema de ruteo de vehículos con flexibilidad en las fechas de entrega.

La política de adelantar pedidos implica la necesidad de almacenar productos por un cierto tiempo. Si bien para el problema real que motivó este trabajo el adelanto permitido sería de un día, con lo cual el costo de almacenamiento sería controlado y aceptable por parte de la compañía, en casos generales donde se permita un mayor adelanto este costo debe ser tenido en cuenta.

Es por ello que en esta tesis se considera el tratamiento de este problema desde diferentes enfoques: mono-objetivo al considerar solo los costos de ruta y multi-objetivo al considerar tanto los costos de ruta como los de almacenamiento.

Para el caso mono-objetivo, se proponen dos formulaciones matemáticas para modelar el problema considerando sólo el objetivo de minimizar la distancia total recorrida. La primera controla el número total de rutas en todo el horizonte de planeación mas no el número de vehículos utilizados cada día. En este sentido la segunda formulación es más adecuada en general, sobre todo en problemas restringidos por la cantidad de vehículos. Las dos formulaciones consideran como parámetro los días que se permite adelantar pedidos, esto es, no sólo modelan el caso del adelanto un día, sino el caso general. Estos modelos pueden ser aplicados a otros problemas donde sea posible aplicar la política de adelantar pedidos.

La demostración de que el problema es NP-completo, así como la complejidad computacional observada al tratar de resolver el modelo de forma exacta nos motivaron a desarrollar metodologías de solución basadas en técnicas metaheurísticas. La primera metodología que se presenta es un método híbrido de GRASP con encadenamiento de trayectorias. La segunda está basada en un método de búsquedas por entornos variables.

Para el enfoque biobjetivo del problema se propone y se valida un modelo matemático que considera la minimización de los costos totales de transportación

y el almacenamiento simultáneamente. Se proponen dos estrategias de solución, la primera de ellas basada en búsquedas por entornos variables enlazadas insertada en un procedimiento de memoria adaptativa para optimización multi-objetivo y la segunda en un método evolutivo.

CONTRIBUCIONES Y CONLUSIONES: La contribución del presente trabajo se centra principalmente en la propuesta de modelación y metodologías de solución de un problema de ruteo de vehículos nuevo al considerar la política de permitir flexibidad en las fechas de entrega de los productos. Otra contribución importante es el estudio del problema desde un enfoque biobjetivo. En la literatura no se reportan trabajos donde se aborde el problema aquí planteado.

De igual forma las metodologías propuestas para su solución resultan novedosas y de acuerdo a los experimentos computacionales realizados, son eficientes y robustas. La implementación computacional de los algoritmos propuestos también se considera como contribución del presente trabajo.

Firmas de los asesores:

Dra. Ada Margarita Álvarez Socarrás

Dr. Joaquín Pacheco Bonrostro

Capítulo 1

Introducción

1.1 Descripción del problema

El presente trabajo tuvo como detonante inicial la problemática que presenta una empresa de repostería ubicada al norte de España, que cuenta con una flotilla de vehículos para transportar sus productos desde un depósito central hacia sus delegaciones. Cada delegación cuenta con una demanda pre-establecida, conocida con certeza con anticipación y requiere los productos un día de la semana que también es conocido. La empresa desea diseñar las rutas de tal forma que se minimicen los costos de transportación y se satisfagan las demandas de las delegaciones en el día de la semana señalado para cada uno de ellos.

Actualmente la compañía hace un diseño de rutas día a día, sin embargo, los responsables logísticos de la empresa observaron que las rutas diarias desde el depósito a las delegaciones daban lugar a costos muy altos y la mayoría de las ocasiones se enviaban vehículos con muy baja ocupación. Cada pedido es entregado por un solo vehículo para que los operarios encargados de descargar el vehículo realicen esta operación una sola vez y el pedido se entrega exactamente el día solicitado por cada delegación. Los vehículos parten del depósito central y retornan a él el mismo día, después de haber finalizado su recorrido.

Entonces la empresa repostera desea realizar un diseño óptimo de sus rutas que satisfagan las demandas de las delegaciones, minimizando los costos de trans-

portación y tratando de reducir el espacio vacío en los vehículos; esto conducirá a un ahorro del número de vehículos utilizados.

Una posible forma de abordar este problema de manera novedosa y que conduzca a ahorros sensibles para la empresa es aprovechar el hecho de que la entrega no se hace a usuarios finales, sino a distribuidores ó delegaciones por lo que el problema pudiera ser replantedo considerando flexibilidad en las fechas de entrega de los productos, o sea, permitiendo que las demandas sean satisfechas en días anteriores a la fecha inicialmente propuesta por la delegación. Debido a que al final de cada semana se conocen con certeza las demandas para cada delegación para la siguiente semana, se considerará un horizonte de planificación de una semana. Esta forma de abordar el problema fue aceptada por la empresa; el adelanto de pedidos no repercute en gastos de producción extra ya que se está trabajando sobre pedidos conocidos y se puede distribuir la producción a lo largo de toda la semana para satisfacer todos los pedidos a tiempo para su envío.

Para que los vehículos puedan visitar mayor cantidad de clientes y transportar una mayor cantidad de productos, proponemos reasignar, de ser posible, el día en que se realizan los envíos, adelantando uno o varios días la fecha de entrega. Como se manejan productos alimenticios, se propone el adelanto de un día para la empresa repostera.

Con lo expresado anteriormente se lograrían disminuir los costos de la empresa, sin embargo, viendo el problema de forma general, permitir un adelanto de varios días la entrega de un producto realmente ocasiona un costo adicional por el almacenamiento de los productos, los cuales deben ser considerados en el costo total. Por consiguiente un estudio más completo del problema implicaría su estudio desde la perspectiva multi-objetivo, buscando soluciones que representen un compromiso entre el costo de transportación y el costo de almacenamiento.

Por las características geográficas de la zona, la ubicación del depósito central, así como de las delegaciones o centros de distribución de la empresa repostera, mini-

mizar la distancia total recorrida repercute directamente en el número de vehículos necesarios para transportar los productos. El depósito se ubica a cierta distancia de donde se encuentran los centros de distribución, por lo que la minimización de la distancia total recorrida incide directamente en la minimización de los costos de transportación. Consideraremos en este trabajo el objetivo de la minimización de los costos de transportación como la minimización de la distancia total recorrida por los vehículos. Asimismo, los costos de almacenamiento están relacionados con el total de productos almacenados, por lo que el objetivo respecto al almacenamiento será la minimización del total de productos que es necesario almacenar al adelantar pedidos.

El problema consiste entonces en diseñar las rutas de toda la semana u horizonte de planeación minimizando la distancia total recorrida y el total de productos que es necesario almacenar al adelantar pedidos. Los supuestos que tenemos son los siguientes:

- La demanda de cada delegación se conoce con certeza con anticipación al igual que el día que cada delegación requiere los productos;
- Se cuenta con una flota homogénea de vehículos;
- Las demandas a cada delegación deben ser satisfechas por un solo vehículo; y
- Cada vehículo inicia y finaliza su recorrido en el depósito el mismo día.

Este problema pertenece a la familia de los problemas de Ruteo de Vehículos (VRP por sus siglas en ingés), por lo que el trabajo se centra principalmente en esta área.

1.2 Antecedentes

El problema de ruteo de vehículos (VRP) es considerado como uno de los problemas más importantes de Optimización Combinatoria. Consiste en diseñar las ru-

tas óptimas para la distribución de productos de un centro de distribución ó depósito a un determinado número de clientes dispersos geográficamente, disponiendo de una flotilla de vehículos con una cierta capacidad de transportación. Cada ruta es realizada por un solo vehículo que inicia y termina en el depósito, de tal forma que se satisfagan los requerimientos de los clientes y las restricciones operacionales; generalmente se busca minimizar el costo global de transportación.

El VRP fue originalmente propuesto por Dantzing y Ramser [26] en 1959 para la distribución de gasolina a estaciones de servicio, proponiendo la primera formulación matemática y un algoritmo de solución aproximada. Desde entonces ha sido objeto de estudio y de investigación, se han propuesto diversas formulaciones matemáticas, algoritmos exactos, pero sobre todo numerosas técnicas heurísticas. La necesidad de resolver variantes más complejas e instancias cada vez mas grandes, han contribuido al desarrollo, estudio y mejora de algunas metaheurísticas [24].

Esta clase de problemas tienen muchas aplicaciones prácticas, [87], como: la distribución de productos alimenticios, entrega de periódicos, recolección de desechos sólidos, asignación de transportes escolares, entre otros. Disminuir los costos de distribución de productos contribuye a reducir los costos de transportación y si se considera que el proceso de transportación que involucra todas las etapas de los sistemas de producción y distribución representa, generalmente, del 10 al 20 % del costo final del producto [87], entonces esta es quizá una de las razones por las que ha tenido tanto auge esta área de la investigación de operaciones.

El VRP es NP–completo, ya que incluye el problema del agente viajero (TSP por sus siglas en inglés) como caso especial, pero en la práctica el VRP es más difícil de resolver que el TSP, por ejemplo, utilizando algoritmos avanzados de ramificación y corte se pueden resolver instancias del TSP con cientos e incluso miles de puntos, en contraste, los algoritmos exactos más sofisticados para el VRP sólo pueden resolver instancias de alrededor de cien puntos y con una tasa de éxito muy variable [61]. Esto explica el porqué mucha investigación se ha concentrado en el estudio de técnicas heurísticas. Otra razón es el hecho de que las heurísticas tienden a ser

considerablemente más flexibles que los algoritmos exactos y pueden adaptarse más fácilmente a diversas variantes que aparecen en la práctica.

Entre las aplicaciones recientes encontradas en la literatura podemos mencionar por ejemplo el trabajo de Angelelli y Speranza [6], quienes desarrollan un modelo periódico para la recolección de papel para reciclar y el estudio de Gonçalves y Ochi [48] quienes utilizan la metaheurística GRASP para resolver un problema periódico relacionado con la extracción de hidrocarburos en Brasil. Por otra parte Francis et al. [39] trabajan una variante del problema de ruteo de vehículos periódico, llamado PVRP con elección de servicio (PVRP Service Choice) en donde consideran como decisión la frecuencia del servicio; de tal forma que se obtiene un beneficio al visitar un cliente en un periodo dado. Proponen una formulación matemática y una heurística de solución.

En los últimos años se ha dado un gran impulso a la modelación de problemas de decisión utilizando múltiples objetivos. Los trabajos publicados sobre problemas de ruteo multi-objetivo son de fechas muy recientes. Entre estos destacan las aportaciones de Jozefowiez et al. [54] y [55] con la propuesta de nuevas técnicas de optimización multi-objetivo aplicadas a problemas de ruteo de vehículos, quienes se plantean como objetivos la minimización de los costos así como lograr el balanceo de rutas, esto es, la diferencia entre las longitudes de la ruta más larga y la más corta.

Otros trabajos interesantes son por ejemplo el de Feillet et al. [35] quienes presentan una descripción del problema del agente viajero con beneficio (TSPP: TSP with profits) el cual es una generalización del TSP donde no es necesario visitar todos los nodos del grafo y donde cada nodo tiene asociado un beneficio al ser visitado. Se puede formular como un problema biobjetivo donde se busca maximizar el beneficio de acuerdo a los clientes visitados y minimizar los costos de transportación. La extensión del TSPP considerando varios vehículos se conoce como VRP con beneficios.

Aksen y Aras [3] abordan un problema de ruteo de vehículos con selección de clientes y maximización de beneficios, este es definido como una generalización

del VRP donde cada cliente tiene una fecha con requerimiento de productos y un beneficio al ser visitado; no es necesario visitar a todos los clientes. El objetivo que se persigue es encontrar el número y las rutas de vehículos que satisfaga las restricciones del tiempo de entrega de productos y maximice el beneficio total.

Wen et al. [89] presentan un problema de ruteo de vehículos multi-periódico dinámico (Dynamic multi-period VRP), en donde los clientes ordenan los productos de forma dinámica sobre un horizonte de planeación que consiste de varios períodos o días. Se desea minimizar los costos de transportación y el tiempo de espera de los clientes y balancear la carga de trabajo diaria sobre el horizonte de planeación.

Pacheco y Martí [76], Corberán et al. [21], Delgado y Pacheco [28], Alabas-Uslu [4] resuelven un problema de dos objetivos de transporte escolar, minimizando número de vehículos y longitud de ruta mas larga, utilizando metaheurísticas.

Dentro de la literatura reciente sobre problemas de ruteo con aplicaciones a distribución de productos perecederos, podemos mencionar los trabajos desarrollados por Eglese et al. [30], Hsu et al. [52], Osvald et al. [75], Tarantilis et al. [85], Ambrosino et al. [5] y Naso et al. [72]. En la mayoría de los casos se modela el problema como un VRP con ventanas de tiempo (VRP-TW). En el problema que estamos trabajando no aplica esta modelación ya que tenemos la condición de que los vehículos deben partir y regresar al depósito en un solo día y el VRP-TW no garantiza esta restricción. Cabe mencionar, que en la literatura reciente no pudimos encontrar trabajos relacionados con la politica de adelanto de pedidos y que busquen minimizar los costos de transportación y de almacenamiento simultáneamente, que sean abordados como se propone en esta tesis.

1.3 Objetivos

Los objetivos que se persiguen en esta tesis son:

- Estudio de un problema novedoso de rutas que está basado en una problemática

real.

- Estudio de este problema desde una perspectiva mono-objetivo considerando la políitica de adelanto de pedidos minimizando la distancia total recorrida.
- Diseño de algoritmos de solución para el problema mono-objetivo basados en estrategias metaheurísticas y evaluación del beneficio que reporta la política de adelanto de pedidos.
- Estudio del problema más general, con un enfoque biobjetivo al considerar la minimización de la distancia total recorrida y el almacenamiento.
- Diseño de algoritmos de solución para el problema biobjetivo basados en estrategias metaheurísticas.
- Implementación de los algoritmos en un software especializado y realización de experimentos computacionales para medir el desempeño de los algoritmos propuestos.

1.4 Contribución científica

La contribución científica esperada con el trabajo se resume en los siguientes puntos:

- Modelación matemática del problema de ruteo con flexibilidad en las fechas de entrega tanto desde la perspectiva mono-objetivo como biobjetivo.
- Diseño de algoritmos eficientes para dar solución a los problemas formulados.

1.5 Relevancia

La relevancia del trabajo está dada por dos cuestiones fundamentales. Por un lado, parte del trabajo desarrollado contribuye a dar solución a una problemática

real. El problema de minimizar la distancia total recorrida y el almacenamiento no ha sido abordado como tal en la literatura científica, por lo que el estudio desarrollado, los modelos propuestos y los algoritmos diseñados pueden ser aplicados eficientemente en situaciones similares donde se permita adelantar pedidos.

Por otra parte el estudio más general del problema desde la perspectiva multi-objetivo, permitirá contar con metodologías adecuadas para resolver problemas similares dentro del área de ruteo de vehículos, así como que sentará las bases para poder considerar dentro de este tipo de problemas otros objetivos importantes.

1.6 Estructura de la tesis

El documento está estructurado de la siguiente forma: en el capítulo dos se describirá el marco teórico con los conceptos necesarios para el estudio del problema. Se describen las características de los problemas de ruteo de vehículos, el modelo clásico VRP capacitado, su formulación matemática y algunas de sus variantes más importantes; se comentarán algunos métodos de solución tratados en la literatura, asimismo, se describirán los problemas de ruteo multi-objetivo que se han documentado recientemente y sus técnicas de solución.

En el capítulo tres se presenta el planteamiento del problema desde el enfoque de un solo objetivo: minimizar la distancia total recorrida; se presentan dos modelos matemáticos lineal entero mixto y su validación con software comercial. El problema es NP-completo, la demostración se presenta en este capítulo. Finalmente, se presenta la relación del problema con otros modelos descritos en la literatura para problemas de ruteo de vehículos.

En el capítulo cuatro se describen dos propuestas de metolodogías de solución para el problema con un solo objetivo, ambas basadas en métodos metaheurísticos. La primera es un método híbrido de GRASP con encadenamiento de trayectorias y la segunda está basada en búsquedas por entornos variables. Ambas metodologías

se describen a detalle.

En el capítulo cinco se presenta el problema desde un enfoque biobjetivo, se propone un modelo matemático lineal entero mixto que considera la minimización de los costos de transportación y del almacenamiento simultáneamente. Se describen algunos elementos importantes de la optimización multi-objetivo necesarios para abordar el problema y se presenta la validación del modelo resolviendo instancias pequeñas aplicando algunos métodos clásicos de optimización multi-objetivo.

En el capítulo seis se presenta la propuesta de dos metodologías de solución para el modelo biobjetivo con técnicas metaheurísticas. La primera de ellas está basada principalmente en el método MOAMP acoplado con una búsqueda por entornos variables, adaptada a partir del algoritmo para un solo objetivo. La segunda metodología se inspira en en el algoritmo NSGA-II, uno de los algoritmos evolutivos más populares para resolver problemas multi-objetivo reportados en la literatura, se diseñaron operadores específicos que se ajustan a las condiciones del problema.

En el capítulo siete se muestran algunos resultados de experimentos computacionales realizados para evaluar el desempeño de los algoritmos de solución propuestos.

Finalmente en el capítulo ocho se presentan las conclusiones y propuesta de trabajo a futuro.

Capítulo 2

Marco teórico

Presentamos en este capítulo las definiciones y algunos resultados necesarios para abordar el problema que estamos trabajando y que se necesitarán en todo el documento. El problema que nos interesa resolver está dentro de los llamados problemas de ruteo de vehículos (VRP, por sus siglas en inglés), por lo que es necesario estudiar las características de este tipo de problemas y los métodos de solución reportados en la literatura para resolverlos.

En la primera sección se presentan los conceptos básicos sobre el problema de ruteo de vehículos, el modelo clásico así como una descripción de algunos problemas de ruteo importantes. También se discuten aspectos relevantes de problemas de ruteo multi-objetivo. La sección 2 está dedicada a la descripción de los principales métodos de solución existentes reportados en la literatura, métodos exactos y técnicas heurísticas y metaheurísticas, tanto para problemas con un solo objetivo como con múltiples objetivos. Se menciona su aplicación a diversos problemas de ruteo.

2.1 El problema de ruteo de vehículos

El problema de ruteo de vehículos consiste en diseñar las rutas óptimas para la distribución de productos de un centro de depósito a un determinado número de clientes dispersos geográficamente, disponiendo de una flotilla de vehículos con una cierta capacidad de transportación. A cada ruta se le asigna un único vehículo que inicia y termina su recorrido en el depósito, de tal forma que se satisfagan los

requerimientos de los clientes y las restricciones operacionales; generalmente se busca minimizar el costo global de transportación que involucra la distancia total recorrida, número de vehículos, entre otros factores.

Algunos documentos que presentan el estado del arte sobre problemas de ruteo de vehículos son Toth y Vigo [87], Cordeau et al. [23] y [24], Mester y Bräysy [70], Pisinger y Ropke [78] y Prins [80], entre otros.

Describiremos enseguida las componentes típicas de un problema de ruteo.

La red vial, usada para la transportación de los bienes, generalmente es descrita por un grafo, cuyos arcos representan secciones o tramos viales y cuyos vértices corresponden a las intersecciones carreteras que son las localizaciones de los clientes y el depósito. Los arcos pueden ser dirigidos o no dirigidos dependiendo de si se pueden recorrer en una sola dirección o ambas. Cada arco tiene asociado un costo, que generalmente representa su longitud o tiempo de viaje.

Las características típicas de los clientes son:

- Un vértice en la red vial que representa su localización.
- Demanda o cantidad de bienes (pedidos) que deben ser entregados o recolectados.
- Ventanas de tiempo: período durante el cual el cliente debe ser atendido.
- Tiempo necesitado para entregar o recolectar los bienes o pedidos.
- Conjunto de vehículos disponibles que pueden utilizarse para atender al cliente.

En algunas ocasiones no es posible satisfacer enteramente las necesidades de cada cliente. En estos casos, se reducen las cantidades de bienes a entregar o recolectar o un subconjunto de clientes puede no ser atendido. En estos casos, usualmente, se asocian diferentes prioridades o penalidades a cada cliente.

Los bienes se transportan en vehículos, las características típicas de los vehículos son:

- Depósito de los vehículos, posibilidad de finalizar el servicio en un depósito distinto al origen.
- Capacidad del vehículo, expresado como peso máximo, volumen, número de pallets, y carga del vehículo.
- Compartimentos o subdivisión del vehículo, cada uno caracterizado por su capacidad y por el tipo de bienes que puede transportar.
- Dispositivos disponibles para las operaciones de carga y descarga.
- Subconjunto de arcos del grafo por los que el vehículo puede transitar.
- Costo asociado a la utilización del vehículo, en unidades de distancia, de tiempo, por ruta, etc.

Los objetivos típicos en un problema de ruteo de vehículos son los siguientes:

- Minimización del costo global de transportación, que depende de la distancia recorrida y costos fijos asociados a cada vehículo.
- Minimización del número de vehículos necesarios para satisfacer a todos los clientes.
- Balanceo de rutas para la carga de los vehículos o del tiempo de viaje.
- Minimización de las penalidades asociadas con el servicio parcial a los clientes.

En la práctica usualmente para la formulación de un problema particular se toma como base la formulación del problema de ruteo capacitado considerado como el problema clásico, el cual puede adaptarse a diferentes situaciones que dan lugar al problema específico que se quiera tratar. Enseguida presentamos la versión básica del problema de ruteo de vehículos capacitado (CVRP).

2.1.1 El problema de ruteo de vehículos capacitado

La versión básica del VRP es el problema capacitado, en el que todos los vehículos tienen la misma capacidad [87]. En general, el VRP se puede modelar como un problema de flujo en redes, con variables binarias, con una gran cantidad de restricciones, entre otras, una por cada subconjunto del conjunto de nodos. Cada cliente o centro de depósito se considera como un nodo de una red y se tienen arcos que unen estos nodos. Asociado a cada arco se tiene un costo no negativo que representa el costo del envío de unidades entre dos nodos de la red. Cada cliente tiene una demanda de productos que desea recibir y se cuenta con un número K de vehículos disponibles. Cada uno de ellos cuenta una capacidad limitada de carga Q. Se desea determinar las rutas de distribución desde el depósito a los clientes de tal forma que se minimicen los costos y se satisfagan las demanda.

El CVRP consiste en encontrar una colección de exactamente K ciclos, cada uno de ellos que corresponde a una ruta de un vehículo, con mínimo costo. Se define el costo total como la suma de los costos de los arcos que pertenecen al ciclo y tal que

i) Cada ciclo visita el depósito,

ii) Cada cliente es visitado exactamente por un ciclo, y

iii) La suma de las demandas de los vértices de un ciclo no exceda la capacidad del vehículo Q

Formulación matemática

Consideraremos que contamos con un problema donde

- $G = (V, A)$ grafo completo no dirigido.
- $V = \{v_0, v_1, v_2, \ldots, v_n\}$ conjunto de nodos, donde v_0 es el depósito.

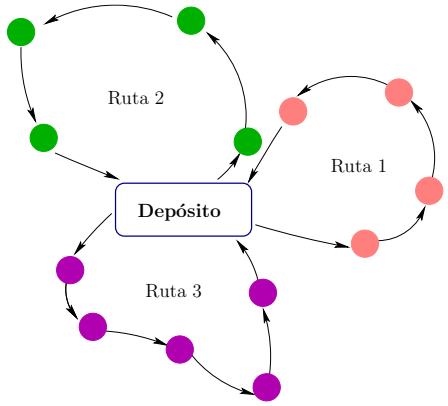


Figura 2.1: Ejemplo de solución de un VRP

- $A = \{(i, j) : i, j \in V, i \neq j\}$ conjunto de aristas.
- C matriz de costos sobre el conjunto de aristas donde c_{ij} representa el costo no negativo asociado al ir del vértice i al vértice j. Esta matriz es simétrica.
- La matriz de costos satisface la desigualdad triangular, esto es

$$c_{ij} \leq c_{ik} + c_{kj}$$

  para todos los nodos del grafo.
- K vehículos iguales con capacidad limitada Q.
- Cada nodo cuenta con una demanda conocida d_i.

Usualmente se denota como 0 el nodo depósito y los vehículos parten y finalizan en él después de realizar el recorrido. Entonces el problema es encontrar K ciclos que parten del nodo depósito, recorren algunos nodos y regresan al nodo depósito, con mínimo costo.

El modelo de dos índices, tomado de [87], considera

$$x_{ij} = \begin{cases} 1, & \text{si la solución utiliza el arco } (i, j) \\ 0, & \text{en otro caso} \end{cases}$$

La formulación matemática asociada al problema es la siguiente.

$$\text{mín} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

sujeto a

$$\sum_{i \in V} x_{ij} = 1, \ \ \forall j \in V \backslash \{0\} \tag{2.1}$$

$$\sum_{j \in V} x_{ij} = 1, \ \ \forall i \in V \backslash \{0\} \tag{2.2}$$

$$\sum_{i \in V} x_{i0} = K \tag{2.3}$$

$$\sum_{j \in V} x_{0j} = K \tag{2.4}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \ \ \forall S \subset V \backslash \{0\}, \ \ S \neq \emptyset \tag{2.5}$$

$$x_{ij} \in \{0, 1\}, \ \forall i, j \in V \tag{2.6}$$

Las restricciones (2.1) y (2.2) imponen la existencia de un solo arco entrante y un solo arco saliente para cada uno de los clientes, esto es, que cada cliente es servido por uno y sólo un vehículo. Análogamente, las restricciones (2.3) y (2.4) imponen el grado para el vértice depósito, los K vehículos salen y llegan a este nodo.

Por último, la restricción (2.5) impide la existencia de subtours, éstas son llamadas restricciones de capacidad y corte, las cuales imponen tanto la conectividad de la solución como los requerimientos de capacidad de los vehículos, aquí $r(S)$ es el número mínimo de vehículos necesarios para satisfacer la demanda en S. Estas restricciones tienen cardinalidad que crece exponencialmente con n, lo cual dificulta mucho la resolución, en [87] se sugiere sustituirlas por una familia de restricciones con cardinalidad polinomial, dadas por

$$u_i - u_j + Q x_{ij} \leq Q - d_j \ \ \forall i, j \in V \backslash \{0\}, \ i \neq j, \ \text{tal que } d_i + d_j \leq Q$$

$$d_i \leq u_i \leq Q, \ \ \forall i \in V \backslash \{0\}$$

donde u_i, $i \in V \backslash \{0\}$, son variables continuas adicionales que representan la carga del vehículo después de visitar al cliente i.

En caso de contar con un solo vehículo y capacidad ilimitada en los vehículos, tendremos el clásico problema del agente viajero (TSP), que se puede ver como un caso particular del VRP.

2.1.2 Algunos problemas de ruteo importantes

Mencionaremos enseguida algunos de los problemas de ruteo más reportados en la literatura.

VRP con ventanas de tiempo

El problema de ruteo con ventanas de tiempo (VRP with Time Windows VRPTW) es una de las variantes más importantes del problema clásico, en el cual el servicio en cada cliente i debe iniciar en un período de tiempo dado $[a_i, b_i]$ llamado ventana de tiempo. El vehículo puede llegar antes de a_i pero tendrá que esperar a que el cliente esté listo para ser atendido, pero no podrá llegar después del tiempo b_i. El tiempo de servicio en el cliente i se denota por s_i.

VRPTW consiste en determinar K ciclos con costo mínimo tal que

i) Cada ciclo visita el depósito.

ii) Cada cliente es visitado exactamente por un ciclo.

iii) La suma de las demandas de los vértices de un ciclo no exceda la capacidad del vehículo Q.

iv) Para cada cliente i el tiempo de servicio inicia en el período $[a_i, b_i]$ y el vehículo se detiene por s_i tiempo.

VRPTW es NP-completo ya que generaliza el CVRP cuando $a_i = 0$ y $b_i = \infty$ para cada cliente i.

La relación del VRPTW con el problema abordado en este trabajo se detalla en la sección 3.4.

VRP con *Backhauls*

El problema de ruteo con *Backhauls* (VRPB) es una extensión del CVRP en donde el conjunto de clientes se particiona en dos subconjuntos, el primer subconjunto L contiene n clientes *Linehaul* que requieren recibir una cantidad de productos. El segundo subconjunto de clientes B, contiene m clientes *Backhaul*, donde se debe recolectar cierta cantidad de productos. Se tiene una restricción de precedencia entre clientes *Linehaul* y *Backhaul*: cuando una ruta da servicio a los dos tipos de clientes, todos los cientes *Linehaul* deben ser atendidos antes de dar servicio a los clientes *Backhaul*. Cada cliente i cuenta con una demanda no negativa, d_i que será entregada o recolectada dependiendo del tipo de cliente.

El VRPB consiste en determinar K ciclos de costo mínimo tal que

i) Cada ciclo visita el depósito.

ii) Cada cliente es visitado exactamente por un ciclo.

iii) El total de las demandas de los clientes *Linehaul* y *Backhaul* visitados por un ciclo no excede la capacidad del vehículo Q.

iv) En cada ciclo todos los clientes *Linehaul* preceden a los clientes *Backhaul*.

VRPB también es NP–completo, generaliza el CVRP cuando $B = \emptyset$.

VRP con entrega y recolección

En el problema de ruteo con entrega y recolección (VRP Pick and Delivering VRPPD) cada cliente i tiene asociada dos cantidades d_i y p_i que representan la demanda que será entregada y recolectada, respectivamente. Para cada cliente i, $\mathcal{O}_i$

denota el vértice que es el origen de la demanda entregada y $\mathcal{D}_i$ denota el vértice destino de la demanda recolectada.

EL VRPPD consiste en encontrar K ciclos de costo mínimo, tales que

i) Cada ciclo visita el depósito.

ii) Cada cliente es visitado exactamente por un ciclo.

iii) La carga del vehículo durante el ciclo debe ser no negativa y que no exceda la capacidad del vehículo Q.

iv) Para cada cliente i, el cliente $\mathcal{O}_i$ diferente del depósito, debe ser atendido en el mismo ciclo antes del cliente i.

v) Para cada cliente i, el cliente $\mathcal{D}_i$ diferente del depósito, debe ser atendido en el mismo ciclo después del cliente i.

VRPPD es NP–completo ya que generaliza a CVRP cuando $\mathcal{O}_i = \mathcal{D}_i = 0$ y $p_i = 0$ para cada $i \in V$.

VRP Periódico

El VRP Periódico (PVRP) es una generalización del VRP clásico en el que las rutas deben diseñarse sobre múltiples días o períodos, esto es, en un horizonte de planeación digamos de p días. Cada cliente requiere n_i visitas durante el horizonte de planeación distribuidas en posibles calendarios factibles para cada cliente, un calendario es una colección de días en el horizonte de planeción en los cuales los clientes recibirán el servicio. Asignar a un cliente a un calendario implica que el cliente recibirá el servicio en cada día del calendario. Por ejemplo, en un horizonte de una semana con 5 días disponibles, si un cliente requiere dos visitas durante la semana, las combinaciones disponibles pueden ser solamente Lunes-Viernes o Lunes-Jueves o Martes-Viernes, pero no se aceptan otras combinaciones para visitar a este cliente.

PVRP consiste en determinar K ciclos en un horizonte de p días con costo mínimo tal que

i) Cada ciclo visita el depósito.

ii) Cada cliente es visitado por n_i ciclos, donde cada visita se realiza en una combinación de días de visitas disponibles para cada cliente.

iii) La suma de las demandas de los vértices de un ciclo no exceda la capacidad del vehículo Q.

PVRP es NP–completo ya que generaliza a CVRP cuando $p = 1$ y $n_i = 1$ para cada $i \in V$.

La relación del PVRP con el problema abordado en este trabajo se detalla en la sección 3.4.

2.1.3 Problemas de ruteo multi-objetivo

Se reportan en la literatura pocos trabajos relacionados con el planteamiento y solución de problemas de ruteo de vehículos con múltiples objetivos. Jozefowiez et al. [55], presentan un panorama de la investigación en problemas de ruteo con varios objetivos, enfatizando el hecho de que la mayoría de los problemas prácticos involucran diferentes objetivos, pero casi toda la investigación del VRP se centra en problemas con un solo objetivo. En los últimos años han tenido gran interés los problemas de ruteo multi-objetivos, esta área puede considerarse un área reciente ya que el número de publicaciones es moderada; sólo tres publicaciones se reportan antes de 1990, todas las demás han sido publicadas después de 1995 y más de la mitad de las publicaciones aparecieron después del año 2000.

También se comenta el hecho de que, a diferencia de los problemas de ruteo con un solo objetivo donde se tiene un modelo estándar, a saber el CVRP, en el caso

de manejar varios objetivos no se cuenta con un modelo base de cual partir para adecuarlo a problemas más complejos.

En [55] se menciona que en los trabajos de ruteo multi-objetivo reportados en la literatura reciente los objetivos principales son algunos de los siguientes.

- Minimizar costos,
- Minimizar longitud de rutas,
- Minimizar la longitud de la ruta más larga,
- Balance que puede ser de carga, de rutas, número de clientes, tiempo de ruta,
- Maximizar la satisfacción de clientes,
- Minimizar el número de vehículos,
- Maximizar la compacidad de las rutas,

entre otros.

Cuando el problema es tratado con un enfoque mono-objetivo se busca optimizar solo uno de ellos. Sin embargo, cuando se usan modelos multi–objetivo se deben optimizar varios de ellos simultáneamente.

De los problemas de ruteo con enfoque multi-objetivo han tenido especial atención aquellos relacionados con variantes del problema del agente viajero con varios objetivos y el tratamiento desde un enfoque multi-objetivo de problemas con ventanas de tiempo, en donde algunas restricciones de las ventanas de tiempo son tratadas como objetivos.

2.2 Métodos de solución para problemas de ruteo de vehículos

En esta sección se presentan algunos de los principales métodos de solución para problemas de ruteo de vehículos que se reportan en la literatura, tanto para problemas con un solo objetivo como con múltiples objetivos. Para problemas de un solo objetivo éstos pueden dividirse en métodos exactos, heurísticos y metaheurísticos. Para problemas multi-objetivo principalmente se utilizan técnicas metaheurísticas.

2.2.1 Métodos de solución para problemas mono-objetivo

Se reportan en la literatura algunos métodos adecuados para la solución de los problemas de ruteo de vehículos con un solo objetivo, éstos se pueden dividir en tres clases: métodos exactos, métodos heurísticos y métodos metaheurísticos. Cuando el número de clientes es muy grande, el número de restricciones crece enormemente y un método exacto tardaría mucho tiempo en encontrar una solución, es por esto que se han desarrollado heurísticas que permiten encontrar soluciones aproximadas a un bajo costo de recursos computacionales. Se ha demostrado en la práctica que estos métodos encuentran muy buenas soluciones de forma eficiente; en los últimos tiempos se han desarrollado más los métodos metaheurísticos con muy buenos resultados.

Dentro de los métodos exactos podemos mencionar el método de ramificación y acotamiento, basado en relajaciones de programación lineal. Otro de los métodos exactos utilizados es el de ramificación y corte, que ha tenido grandes avances en los últimos años, considerándose actualmente el más importante dentro de los métodos exactos [24].

También existen algoritmos heurísticos muy diversos, con distintos grados de complejidad y que utilizan técnicas muy diferentes para abordar el problema. Los algoritmos heurísticos para el VRP pueden dividirse principalmente en dos clases: heurísticas clásicas, diseñadas entre los años 1960 y 1990 y metaheurísticas que

surgieron a principios de los 90 y se han ido desarrollando durante los últimos años. Las heurísticas clásicas realizan una exploración relativamente limitada del espacio de soluciones, proporcionando resultados aceptablemente buenos con tiempos de cómputo moderados. Además, la mayoría de ellas pueden modificarse fácilmente para incorporar nuevas restricciones relacionadas con problemas de la vida real o para acomodarse a variantes del problema. Sin embargo los resultados son de mejor calidad al utilizar técnicas metaheurísticas, que es donde se ha concentrado el desarrollo de nuevos métodos en los últimos años.

Mencionaremos enseguida a detalle los métodos propuestos para el VRP.

Métodos exactos

Los métodos exactos aplicados al problema de ruteo de vehículos capacitado principalmente son los métodos de ramificacón y acotamiento y los de ramificación y corte.

- **Ramificación y acotamiento.** Hasta finales de los 80's los métodos más efectivos eran los algoritmos de ramificación y acotamiento basados en relajaciones combinatorias. En los últimos tiempos se han propuesto relajaciones lagrangianas, las cuales han podido incrementar el tamaño de los problemas que se pueden resolver. Los algoritmos de este tipo se concentran principalmente en la determinación de una buena cota inferior, la cual constituye una de las principales componentes en métodos de este tipo.

  Se han utilizado algunas relajaciones combinatorias, incluyendo las que están basadas en el problema de asignación, problemas de árboles de mínima expansión con restricciones de grado y relajaciones del espacio de estado.

  Algunas referencias de este tema pueden encontrarse en [24], [37] y [87].

- **Ramificación y corte.** Los algoritmos de ramificación y corte tratan de generar desigualdades válidas para añadir planos de corte en el espacio factible de

soluciones, tratando de acotar el árbol de ramificación no sólo en el nodo raíz. Estos algoritmos constituyen los mejores algoritmos exactos hoy en día para el problema de ruteo de vehículos, la investigación en esta área se ha expandido debido al surgimiento exitoso de combinaciones poliédricas. Sin embargo, en un artículo sobre el panorama de la aplicación de métodos de ramificación y corte para el CVRP, Naddef y Rinaldi [71] establecen que *"La cantidad de la investigación utilizada para resolver el CVRP por este tipo de métodos no es comparable con la que se ha dedicado para resolver el TSP, la investigación en este campo es todavía limitada y no se ha publicado"*.

Se ha trabajado con relajaciones lineales [63], estudios poliédricos [25], así como en formulaciones para el CVRP mediante flujos en redes de dos productos [10].

Se ha dado énfasis en el desarrollo de desigualdades válidas, Ropke et al. [82] desarrollan una familia de estas desigualdades para el VRP con entrega y recepción y ventanas de tiempo. Augerat et al. [8] las utilizan con una heurística de separación para el CVRP.

También se ha trabajado la combinación de algoritmos de ramificación y corte con otros algoritmos, Fukasawa et al. [40] proponen un método de solución que combina ramificación y corte con un algoritmo de generación de columnas (Branch and cut and price: BCP por sus siglás en inglés). La idea de combinar columnas y generación de cortes es mejorar las cotas inferiores en el proceso de ramificación. Baldacci et al. [9] proponen un algoritmo basado en partición de conjuntos con cortes adicionales para el CVRP.

Métodos heurísticos

Los algoritmos heurísticos clásicos para el VRP pueden dividirse en tres categorías principales [87]:

- **Algoritmos constructivos.** Construyen gradualmente una solución factible para el problema intentando optimizar la función objetivo, pero no incluyen

ninguna fase de mejora de la solución encontrada.

- **Algoritmos de dos fases.** Descomponen de forma natural el problema en dos etapas, una de agrupación de vértices y otra de construcción de rutas. Estas dos etapas pueden realizarse de forma conjunta o independiente y en cualquier orden.

- **Algoritmos de mejora.** Parten de una solución factible inicial y tratan de mejorarla realizando intercambios de arcos o vértices dentro de cada ruta o entre varias rutas.

Describimos enseguida los detalles de cada uno.

Algoritmos constructivos

Los algoritmos de construcción de rutas fueron los primeros algoritmos para el CVRP, típicamente inician con una solución vacía y de forma iterativa van construyendo rutas insertando uno o más clientes. Estos algoritmos se describen por tres ingredientes principales: un criterio de inicialización, un criterio de selección que especifica qué clientes se escogerán para insertar en la iteración actual y un criterio de inserción el cuál decide dónde localizar al vértice en la ruta actual. Entre otros, podemos mencionar el algoritmo de Clarke–Wright [20] y el de Mole y Jameson [68].

El algoritmo de Clarke-Wright, también conocido como algoritmo de los ahorros, es una de las heurísticas más conocidas para el problema de ruteo de vehículos, fue introducida unos años después de que Dantzing hiciera su primera formulación y logró mejorar su metodología. A pesar de ser un algoritmo *greedy* arrojó importantes resultados para la época y sentó las bases para la formulación de las heurísticas modernas. Su principio fundamental es la fusión de rutas, que se traduce en tomar dos rutas factibles y fusionarlas con el objetivo de obtener una nueva ruta que sea también factible y con un mejor valor de la función objetivo, o sea, un menor costo.

Supongamos de modo general que existen dos rutas, llamémoslas ruta_1 y ruta_2.

La ruta$_1$ sale del nodo depósito denotado por nodo 0, recorre algunos nodos, llega al nodo i y regresa al depósito. De esta forma quedaría definida como $(0, \ldots, i, 0)$. Por otro lado la ruta$_2$ parte del nodo depósito, visita inmediatamente el nodo j, recorre algunos nodos más y termina en el nodo depósito. De esta forma quedaría definida como $(0, j, \ldots, 0)$. Se pretenden fusionar estas dos rutas formando una ruta$_3$ que inicie en el depósito, realice el recorrido de la ruta$_1$ y al llegar al nodo i, en lugar de regresar al depósito, viaje al nodo j y realice el recorrido de la ruta$_2$ hasta llegar al depósito de la siguiente forma $(0, \ldots, i, j, \ldots, 0)$. Habrá fusión de las rutas si y sólo si la ruta$_3$ tiene menor costo que la suma de los costos de la ruta$_1$ y ruta$_2$ antes de fusionar. El ahorro viene dado por

$$s_{ij} = c_{i0} + c_{0j} - c_{ij}$$

Si el ahorro es positivo, será conveniente la fusión. Se tienen versiones secuencial y en paralelo del algoritmo, se realiza la fusión de mayor ahorro en cada iteración y el proceso termina cuando ya no es posible fusionar dos rutas.

Otra heurística constructiva reportada en la literatura es la de Mole y Jameson, la cual utiliza como criterio de selección y de inserción la evaluación de una distancia extra que resulta de insertar un vértice no ruteado l entre dos vértices consecutivos i y j en la ruta actual,

$$\alpha(i, l, j) = c_{il} + c_{lj} - \lambda c_{ij}$$

donde λ es un parámetro controlado por el usuario.

Algoritmos de dos fases

Los métodos de dos fases están basados en la solución de dos subproblemas: 1) agrupamiento, esto es, determinar una partición de los clientes en subconjuntos y 2) ruteo, esto es, determinar la sucesión de clientes en cada ruta. En un método de agrupa primero y rutea después, los clientes primeramente son agrupados en clusters y entonces las rutas se determinan asignando una secuencia adecuada a cada cluster.

Se han propuesto diferentes formas para la fase de agrupamiento, mientras que para la fase de ruteo regularmente se aplican algoritmos del agente viajero.

El algoritmo de barrido (sweep) [42] es un ejemplo de algoritmo de agrupa primero rutea después, se aplica sobre instancias que definen los vértices sobre el plano cartesiano. Se inicia con un cliente arbitrario y se van asignando el resto de los clientes al vehículo considerándolos en orden creciente del ángulo que forman respecto al depósito y el cliente inicial. Si un cliente no se puede asignar factiblemente al vehículo, se genera una nueva ruta con él. Cuando todos los clientes están asignados a los vehículos se aplica a cada ruta un algoritmo para resolver el problema del TSP.

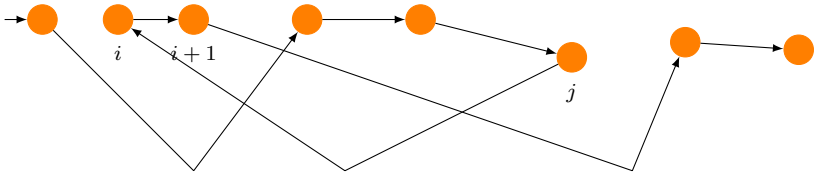
Fisher y Jaikumar [38] proponen una heurística de dos fases, en la primera fase se genera un vértice semilla en la región donde se desee situar la ruta y los clientes se asignan resolviendo un problema de asignación generalizada (GAP), esto es, minimizando la suma de las distancias entre los clientes y las semillas, sujeto a las restricciones de que la demanda total de cada cluster no exceda la capacidad de los vehículos. Se resuelve un TSP para cada cluster.

Algoritmos de mejora

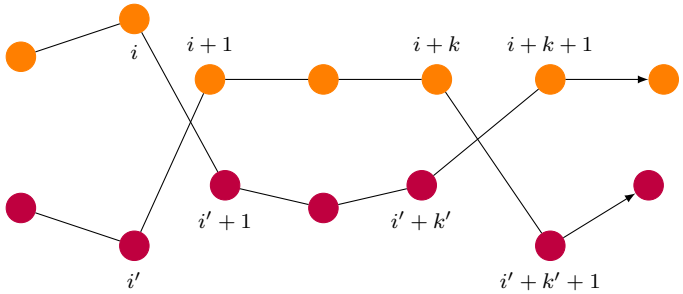
Existen dos formas básicas de mejorar una solución factible encontrada para un problema de ruteo, mediante movimientos intra-rutas [73] o inter-rutas [84].

Los movimientos intra-rutas consisten en mover o intercambiar vértices o cadenas de vértices de una misma ruta. Destacan entre ellos los llamados movimientos tipo *2–opt*, los cuales son muy utilizados en problemas TSP y consisten en intercambiar dos aristas reconectando el tour para dar lugar a uno nuevo de más bajo costo. Se generalizan a movimientos r–opt en los que se intercambian r aristas del tour. Los movimientos inter-rutas consisten esencialmente en mover vértices o cadenas de vértices de una ruta a otra, o como un intercambio de cadenas de vértices en los que ambas rutas entregan y reciben elementos. Se ilustran con la figura 2.2.1.

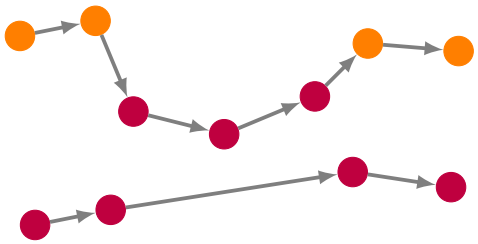
Generalmente la exactitud de las heurísticas clásicas no es muy buena, pero



Movimiento de cadenas de vértices dentro de una misma ruta



Intercambio de cadenas de vértices en rutas diferentes



Movimiento de cadenas de vértices de una ruta a otra ruta diferente

Figura 2.2: Movimientos en la búsqueda local

se siguen utilizando por su simplicidad y porque pueden ser puntos de partida para otros algoritmos. En los últimos 15 años se han desarrollado más los métodos metaheurísticos para resolver problemas de ruteo y se ha tenido un gran avance en esta dirección [61]. Presentamos enseguida una breve descripción de las principales metaheurísticas.

Métodos metaheurísticos

Según Osman y Kelly [74] *Una metaheurística es un procedimiento iterativo, con una estructura y unas reglas generales de funcionamiento que lo caracterizan, que guía un método (normalmente un heurístico) subordinado combinando inteligente-*

mente diversos conceptos para explorar los espacios de búsqueda utilizando estrategias aprendidas para conseguir soluciones quasi-óptimas de manera eficiente

Según Glover y Laguna [46] *Metaheurística se refiere a una estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de aquellas que normalmente se generan en una búsqueda de óptimos locales*

En las metaheurísticas se intenta explorar de una forma más profunda las regiones más prometedoras a priori del espacio de soluciones, combinando búsquedas locales y recombinación de soluciones parciales. Una diferencia importante con las heurísticas clásicas es que pueden aceptar un pequeño empeoramiento de la función objetivo con la intención de conseguir posteriormente un beneficio mayor y escapar de mínimos locales en los cuales pudiera quedar atrapado el algoritmo. La calidad de las soluciones proporcionadas por estos métodos es generalmente bastante mayor que la de las soluciones obtenidas con las heurísticas clásicas, a cambio de un tiempo de cómputo mucho más elevado y una mayor complejidad algorítmica.

Una de las componentes más importantes dentro de las metaheurísticas es la búsqueda local, esto es, la exploración local del entorno de una solución buscando mejores soluciones, se describe enseguida.

Búsqueda local: Una búsqueda local se basa en la idea de explorar las soluciones vecinas de aquella que tenemos en un momento dado. Para diseñar un procedimiento de búsqueda local, es necesario definir previamente un vecindario o entorno $N(S)$:

$$N(S) = \text{Conjunto de soluciones vecinas de } S \text{ a las que se llega por un pequeño movimiento o cambio en } S$$

$$f(S) = \text{Valor de la función objetivo en } S$$

Una búsqueda local inicia con una solución S_0 y se mueve en cada iteración t de una solución S_t de valor $f(S_t)$ a otra solución localizada en la vecindad $N(S_t)$.

En muchos casos, S_t es la solución actual, pero algunos mecanismos multi-arranque permiten reinicializar desde otro punto. La vecindad $N(S_t)$ consiste de

todas las soluciones que pueden alcanzarse desde S_t aplicando una transformación o un movimiento dado. La búsqueda finaliza con la mejor solución encontrada S^* cuando se satisface el criterio de parada, usualmente cuando se llega a un número de iteraciones dado o cuando se tienen varias iteraciones sin mejora en S^*.

Las formas más usuales de seleccionar la solución vecina son explorar todo el vecindario y tomar la mejor solución según el valor de la función objetivo (mayor descenso) o buscar y seleccionar la primera que mejora la solución actual (primer descenso).

Describimos enseguida brevemente algunas de las principales metaheurísticas utilizadas para resolver diversos problemas de ruteo de vehículos.

Búsqueda Tabú

La búsqueda Tabú desarrollada por Glover y Laguna [46] es la principal metaheurística dentro de lo que se conoce como programación mediante memoria adaptativa, que se caracterizan por ser métodos de búsqueda por entornos en los cuales se utiliza, de forma explícita, información acerca de los movimientos realizados con anterioridad. Los principales atributos de cada solución visitada son almacenados en una lista tabú por un determinado número de iteraciones, para evitar que estas soluciones sean visitadas nuevamente, y así evitar ciclos en la búsqueda. Un elemento del vecindario de la solución actual es declarado tabú si alguno de sus atributos está en la lista tabú. Se realiza una búsqueda por entornos en la cual se desplaza en cada iteración a la mejor solución no tabú del vecindario de la solución actual utilizando un criterio determinístico.

Si S_t es la solución actual, entonces S_{t+1} será la mejor solución en $N(S_t)\backslash T(S_t)$, donde $T(S_t)$ es el conjunto de soluciones tabú o prohibidas en la iteración t.

La búsqueda tabú ha sido utilizada exitosamente en la solución de diversos problemas de ruteo, por ejemplo Cordeau et al. en [22] y en [23] presentan la descripción de la solución a problemas de ruteo periódicos y con ventanas de tiem-

po haciendo uso de algoritmos de búsqueda tabú. El algoritmo reportado en [22] está catalogado en la literatura como uno de los más eficientes para resolver problemas de ruteo de vehículos, incluso se utiliza para comparar resultados con nuevos algoritmos.

En VRP un mecanismo tabú puede implementarse como se realiza en [22], en donde se construye un conjunto de atributos para cada solución visitada, se define para la solución actual S_t

$$B(S_t) = \{(i,k) : i \in V\backslash\{0\}, k = 1,\ldots,K\}$$

esto es, se asocia a cada solución los clientes (i) y el vehículo asignado a ese cliente, (k). Una solución vecina consiste en eliminar el par (i,k) de $B(S_t)$ e insertar otro par $(i,k\prime)$ en su lugar. El par (i,k) se declara tabú por un determinado número de iteraciones.

Búsqueda por entornos variables

Otro algoritmo basado en vecindades es el de búsqueda por entornos variables desarrollado por Hansen y Mladenović [49], (VNS: Variable Neighborhood Search). En este algoritmo se definen varias estructuras de vecindades, la búsqueda inicia en una estructura de vecindad definida y al localizar un mínimo local, se procede a la siguiente vecindad; la búsqueda reinicia en la primera vecindad cuando se encuentra una mejor solución S^* o cuando ya se exploraron todas las vecindades. En algunas ocasiones se pueden definir las vecindades anidadas, pero no es requisito para que el método trabaje.

Un bosquejo de un algoritmo de una búsqueda por entornos variables básica es el siguiente:

Algoritmo búsqueda por entornos variables básico

Inicialización: Seleccionar un conjunto de estructuras de vecindad $N_l,\ l = 1,2,\ldots,l_{max}$.

Encontrar una solución inicial S

Repetir

1. $l = 1$
2. Repetir hasta que $l = l_{max}$
   a) Agitación: Generar una solución $S' \in N_l(S)$ de forma aleatoria
   b) Búsqueda local: Aplicar un procedimiento de búsqueda local a partir de S'. Sea S'' el resultado de esta búsqueda local.
   c) Moverse o no:
   Si S'' es mejor que S; $S \leftarrow S''$ y $l = 1$
   De lo contrario $l = l + 1$.

Hasta satisfacer un criterio de paro

Los algoritmos basados en búsquedas por entornos variables también han sido utilizados en resolver problemas de ruteo, entre algunos de los principales trabajos utilizando esta técnica podemos mencionar el de Ergun et al. [33], Braysy [16] para problemas con ventanas de tiempo y Hemmelmayr et al. [51] para problemas de ruteo periódicos.

Las principales formas de definir los vecindarios son con movimientos 2-opt, intercambios de vértices o cadenas de vértices entre rutas o movimientos de cadenas de vértices de una ruta a otra.

Algoritmos genéticos

Los algoritmos genéticos son metaheurísticas clasificadas dentro de los métodos poblacionales, esto es, que involucran una población de soluciones. Las soluciones se consideran como cromosomas y se reproducen al someterse a procesos de cruzamiento o mutación. En el cruzamiento se toman dos soluciones padres de la población y se combinan para generar una o más soluciones hijas. La mutación generalmente es

aleatoria y consiste en cambiar algún elemento de la solución. Todos los elementos son evaluados y se consideran las mejores soluciones para pasar a la siguiente generación.

Dentro de los algoritmo genéticos aplicados en la resolución de problemas de ruteo de vehículos destaca la propuesta de Prins [80]. Prins transforma las soluciones de un VRP en soluciones del TSP eliminando las visitas al depósito por todas las rutas, opera con este tour gigante para cruzamientos y mutaciones y luego, transforma el problema en uno de la ruta más corta para poder obtener soluciones factibles para el VRP. Este algoritmo es citado en la literatura como uno de los algoritmos con mejores resultados para el VRP.

Búsqueda Dispersa

Dentro de las búsquedas por poblaciones podemos mencionar también la búsqueda dispersa (SS: Scatter Search) desarrollada por Glover [43] y [44] y Laguna y Martí [60]. La búsqueda dispersa es un metaheurística evolutiva que opera sobre un conjunto de soluciones, que la literatura refiere como el conjunto de referencia: *RefSet*. La evolución del *RefSet* se logra a través de la combinación de soluciones de referencia para obtener soluciones de prueba. El RefSet es un conjunto de "buenas" soluciones encontradas durante la búsqueda, en este caso "buenas soluciones" no se limita sólo a la calidad de la solución, medida por el valor de la función objetivo. Por ejemplo, una solución puede ser buena, ya que proporciona diversidad con respecto a otras soluciones en el conjunto de referencia.

Algunas implementaciones de SS dividen el *RefSet* en dos subconjuntos que contribuyen con calidad de la solución y diversidad.

Un algoritmo base de SS se compone de las siguientes cinco etapas

1. Se construye un conjunto de soluciones iniciales por un método generador diversificador.

2. Se aplica un método de mejora para modificar las soluciones iniciales.

3. Se escogen soluciones de alta calidad y soluciones diversas por un método de actualización del conjunto de referencia.

4. Un método generador de subconjuntos determina qué soluciones del conjunto de referencia servirán como base para la creación de nuevas soluciones.

5. Se generan nuevas soluciones por un método de combinación.

Algunas aplicaciones a problemas de ruteo son por ejemplo Russel y Chiang [83] y Prado [79] para resolver problemas con ventanas de tiempo.

GRASP: Procedimiento de búsqueda miope, aleatorizada y adaptativo

Esta metaheurística (GRASP por sus siglas en inglés: Greedy Randomized Search Procedure) [36] es un proceso iterativo, donde cada iteración consiste en dos fases: construcción y búsqueda local. En la primera fase se construye una solución inicial factible y, a partir de ésta, se explora una vecindad en la fase de búsqueda local, obteniendo un óptimo local. El resultado es la mejor solución sobre todas las iteraciones.

La fase de construcción de GRASP es un proceso iterativo donde, en cada iteración, los elementos que no pertenecen a la solución parcial se evalúan por una función "miope", la cual estima el beneficio de incluirlos en la solución parcial. Estos elementos se ordenan por su valor estimado y los mejores se incluyen en una lista restringida de candidatos (RCL) y uno de ellos, escogido aleatoriamente, es el que se incluye en la solución actual. El tamaño de la lista candidata se limita por un parámetro dado y el proceso termina cuando se genera una solución factible.

Las soluciones generadas por la fase de construcción no tienen garantía de ser óptimos, por lo que usualmente se aplica un proceso de búsqueda local para mejorar esta solución inicial. Se reemplaza sucesivamente la solución actual con la mejor encontrada hasta que no se encuentren mejores soluciones. El algoritmo de GRASP

tomado de [36] es el siguiente.

Procedimiento GRASP
Entrada de Instancia
Mientras no se satisface criterio de parada
Construir una solución inicial ávido–aleatoria
Realizar búsqueda local a partir de la solución inicial
Actualizar la mejor solución encontrada
fin
Salida: mejor solución encontrada
end

Como hay elementos de aleatoriedad incluidos en la construcción de la solución inicial, esto permite que se construyan soluciones diferentes y se espera que se explore una diversidad de regiones para poder obtener una buena solución. La fase de la construcción de la solución inicial es crucial para el buen desempeño del algoritmo ya que debe poder obtener una variedad de soluciones iniciales para optimizar localmente. El algoritmo de construcción de la solución inicial factible es el siguiente.

Procedimiento Construcción inicial ávido–aleatoria
Solución = {}
Mientras no se tenga una solución
Evaluar los elementos candidatos de acuerdo a la función "gredy"
Hacer una RCL: lista restringida con los mejores candidatos
Seleccionar un elemento aleatorio de esta lista restringida
Solución = Solución $\cup\{s\}$
Adaptar la función Greedy incluyendo s en la solución
fin
Salida: una solución inicial
end

La fase de búsqueda local o algoritmo de mejora, depende del problema concreto que se trabaje, lo que se busca es definir vecindades alrededor de una solución dada y realizar movimientos hasta obtener un óptimo local dentro de esta vecindad.

Algunas aplicaciones de GRASP para problemas de ruteo son por ejemplo la desarrollada por L. Gonçalves et al. [48] para VRP periódicos.

2.2.2 Métodos de solución para problemas multi-objetivo

Como se comentó en secciones anteriores se reportan algunas aplicaciones de técnicas clásicas de optimización multi-objetivo al resolver problemas prácticos, pero se ha dado gran impulso principalmente al desarrollo y aplicación de técnicas metaheurísticas novedosas para problemas con múltiples objetivos.

Los métodos clásicos la mayoría de las ocasiones transforman el problema original ponderando las funciones objetivo y sumándolas para dar lugar a un problema de optimización de un solo objetivo. Dentro de las desventajas de estos métodos se puede mencionar que usualmente no son aplicables a problemas combinatorios, la complejidad computacional crece ya que se requiere resolver múltiples problemas con un solo objetivo y posiblemente cada uno de ellos es NP–completo. Usualmente se combinan la técnica clásica y metaheurísticas para la solución del problema con un solo objetivo, sin embargo, el mayor desarrollo se ha dado en las metaheurísticas para problemas multi-objetivo.

En el artículo de Jozefowiez et al. [55] se menciona que los algoritmos más utilizados para resolver problemas de ruteo multi-objetivo se pueden clasificar en tres grupos: los métodos escalares, los llamados métodos de Pareto y el resto de los algoritmos que no pertenecen a ninguna de estas dos clasificaciones. Los métodos escalares utilizan transformaciones matemáticas como sumas ponderadas lineales. Los métodos de Pareto utilizan la noción de dominancia de Pareto para evaluar la calidad de una solución o para comparar soluciones, éstos son principalmente utilizados en los algoritmos evolutivos que cada vez son más populares. La última

categoría considera los objetivos por separado.

Dentro de los métodos escalares más utilizados está el métodos de sumas ponderadas, donde los objetivos son ponderados por pesos, son sumados para dar lugar a una función escalar; sin embargo, este método tiene varias desventajas. La primera de ellas es la elección adecuada de los pesos de acuerdo a la importancia de cada objetivo, la cual no es una tarea fácil. La otra desventaja principal es que el método no es capaz de encontrar todos los puntos del conjunto de Pareto, ya que requiere convexidad en la región a trabajar para garantizar encontrar todos los puntos no dominados. Aún así se utiliza por su simplicidad y porque se pueden utilizar metaheurísticas ya desarrolladas para problemas con un solo objetivo. Ha sido utilizada por Bowerman et al. [15], Lee y Ueng [64] entre otros.

Otra técnica escalar utiliza la programación por metas, esto es, se escoge una meta que es un punto en el espacio de objetivos y se dirige la búsqueda para minimizar la distancia del punto actual a la meta. La principal dificultad se presenta al definir la meta. Destaca en la literatura el trabajo de Jozefowiez et al [56] que utiliza esta técnica donde define dinámicamente la meta y utiliza búsqueda tabú para llegar a ella. Este algoritmo llamado TAPaS se describe a detalle más adelante.

Otra técnica escalar es el método de la ϵ–restricción. En esta estrategia sólo se optimiza uno de los objetivos y los otros son considerados como restricciones, expresadas como $f_i(x) \leq \epsilon_i$. Al cambiar los valores de ϵ_i obtenemos varias soluciones. Este método también presenta la desventaja de la elección de los valores de ϵ_i para determinar todos los puntos en el frente de Pareto. Esta técnica ha sido utilizada por Coerberán et al. [21] utilizando una búsqueda dispersa para cada problema. También ha sido utilizada por Pacheco y Martí [76] que utilizan búsqueda Tabú, entre otros.

Los Métodos de Pareto utilizan directamente la noción de dominancia de Pareto. Se utiliza principalmente para algoritmos evolutivos y con este esquema ha sido ampliamente utilizado para problemas de ruteo multi-objetivo. Más adelante se describen a detalle algunos de ellos.

Algunos otros métodos no utilizan técnicas escalares ni métodos de Pareto, entre estos destacan Vector Evaluated Genetic Algorithm (VEGA) que fue el primer algoritmo genético propuesto para resolver problemas multi-objetivo. En esta categoría también están los métodos lexicográficos en donde los objetivos tienen un valor de prioridad y los problemas se resuelven en orden de prioridad decreciente. Dentro de esta categoría podemos mencionar los trabajos de Doerner et al. [29] y Keller y Goodchild [58].

Describimos enseguida uno de los métodos metaheurísticos para problemas multi-objetivo utilizado para resolver problemas de ruteo.

TAPaS: Target Aiming Pareto Search

Jozefowiez et al. [56] proponen un método para optimización biobjetivo y lo aplican a problemas de ruteo donde consideran los objetivos de minimizar la distancia y el balanceo de rutas.

Se aplica una búsqueda local l_i a cada solución s_i de un conjunto potencialmente de Pareto, P. Se define una función objetivo específica o_i en cada búsqueda local l_i. La función o_i debe tomar en cuenta que se realizarán varias búsquedas locales de tal forma que dos búsquedas locales no exploren la misma área del espacio de objetivos y, al mismo tiempo, tratar de explorar completamente el área dominada por P para converger al conjunto de Pareto. Para ello el espacio de objetivos se divide en cuatro regiones: el área del espacio de objetivos dominada por P, el área del espacio de objetivos no dominada por P y que no domina a cualquier solución de P, el área del espacio de objetivos dominada sólo por una solución de P y el área de espacio de objetivos dominada por más de una solución en P.

Para cada solución s_i se denota por A_s^i el área del espacio de objetivos dominada sólo por s_i. Si l_i puede generar una solución factible en A_s^i, entonces estaremos mejorando la aproximación del conjunto eficiente sin perder diversificación. Para guiar la búsqueda se propone para cada l_i un punto meta g_i, el cual será un punto

que domina a todos los puntos en A_s^i. Se aplica una búsqueda local para tratar de llegar a g_i; durante la búsqueda se lleva un registro de las soluciones no dominadas visitadas. Cuando terminan todas las búsquedas locales se actualiza el conjunto P considerando todas las soluciones visitadas durante la búsqueda local. Se itera este proceso hasta que no cambia el conjunto de soluciones no dominadas. En este contexto el término búsqueda local se utiliza en un sentido general, en particular en [56] se propone utilizar búsqueda tabú por los buenos resultados que se reportan al aplicarla a problemas de ruteo.

En [54] se propone este método como parte de una estrategia de dos fases. En la primera fase se utiliza un algoritmo evolutivo, concretamente un algoritmo genético (NSGA-II), para generar una aproximación inicial del conjunto de Pareto. Con esto se busca logar un conjunto inicial suficientemente diverso en el espacio de objetivos, de tal forma que en la segunda fase este conjunto sea mejorado con un proceso de intensificación, manteniendo la diversificación. El algoritmo TAPaS constituye la fase de intensificación.

2.2.3 Metaheurísticas generales para problemas multi-objetivo

Dentro de las metaheurísticas desarrolladas para problemas multi-objetivo generales se encuentran Búsqueda dispersa multi-objetivo (MOSS) [13], Procedimiento de búsqueda dispersa y búsqueda tabú para optimización multi-objetivo (SSPMO) [69], Metaheurística multi-objetivo con un procedimiento de memoria adaptativa (MOAMP) [18] y un algoritmo genético de ordenamiento no dominado (NSGA-II) [27]. Estas dos últimas serán aplicadas al problema que nos ocupa por lo que se dará una breve descripción de ellas a continuación.

MOAMP: Metaheurística multi-objetivo con un procedimiento de memoria adaptativa

Caballero et al. [18] describen un método para optimización multi-objetivo basado en una serie de búsquedas tabú enlazadas, llamado MOAMP: Multiobjective Metaheuristic using an Adaptive Memory Procedure. Tiene la característica que la aproximación final del conjunto eficiente puede incluir cualquier solución visitada durante la búsqueda. Este procedimiento básicamente genera una aproximación a la curva de eficiencia enlazando una serie de procedimientos de búsqueda tabú y realizando posteriormente un proceso de intensificación, que consiste en una exploración del entorno de las soluciones obtenidas hasta ese momento. MOAMP se basa en las siguientes ideas:

- El principio de proximidad de puntos eficientes según el cual en un entorno o vecindario de una solución eficiente se puede encontrar otra solución eficiente.
- La solución que minimiza la distancia L_∞ al llamado punto ideal, es también eficiente.

El método está compuesto por tres fases. En las dos primeras fases se genera un conjunto de puntos eficientes inicial enlazando una serie de búsquedas tabú, esto es, el último punto visitado por una búsqueda es el punto de partida de la siguiente. La tercera fase realiza un proceso de intensificación alrededor del conjunto de puntos eficientes encontrados al finalizar la segunda fase.

Este será uno de los métodos base que utilizaremos para abordar nuestro problema, se describen más detalles en el capítulo 6.

Algoritmo genético de ordenamiento no dominado: NSGA-II

Dentro de los algoritmos evolutivos los algoritmos genéticos han sido ampliamente utilizados para resolver problemas multi-objetivo. Uno de los métodos más

populares es NSGA-II (Nondominated sorting genetic algorithm-II) desarrollado por Deb [27]. Se trabaja con una población de soluciones y operadores usuales de los algoritmos genéticos como selección de padres, cruzamiento, mutación, etc. Cada solución tiene asociados dos valores: el rango que representa la calidad de la solución en términos de convergencia al conjunto de Pareto y un valor de aglomeración que corresponde a la calidad en términos de diversificación.

Se reporta en la literatura buenos resultados al aplicar este método a problemas multi-objetivo en general, por lo que se considera uno de los métodos más efectivos para aproximar la curva de eficiencia. Se tienen más detalles de este algoritmo en el capítulo 6.

Capítulo 3

El problema mono-objetivo

En este capítulo se presentan dos formulaciones matemáticas que modelan el problema que estamos tratando. Este consiste, como como se planteó en el capítulo 1, en diseñar las rutas en todo el horizonte de planeación, buscando minimizar la distancia total recorrida y satisfaciendo la demanda de los distribuidores en el día señalado ó unos pocos días antes. Se trabajará en este capítulo el problema con un solo objetivo, esto es, se desea minimizar la distancia total recorrida.

En la primera sección se presenta una formulación preliminar del problema, basada en el modelo estándar de dos índices para el VRP capacitado [87], añadiendo las restricciones relacionadas con la flexibilidad en las fechas de entrega. En este modelo se considera el número de rutas total en todo el horizonte de planeación, esto es, no se limita el número de vehículos en cada día. Esto puede representar una desventaja en ciertas aplicaciones donde este parámetro sea muy limitado. Para contrarrestar esta situación se propone una segunda modelación en donde se controla el número de vehículos en cada día del horizonte de planeación.

El segundo modelo se describe a detalle en la segunda sección. Este puede verse como un modelo intermedio entre el VRP capacitado y el VRP periódico. A diferencia del primero, en este modelo se controla el número de vehículos en cada día del horizonte de planeación, agregando algunas restricciones y variables para este fin.

Se validan ambas formulaciones resolviendo de forma exacta en instancias

pequeñas con software comercial.

De las formulaciones propuestas y su resolución exacta en algunas instancias de prueba, se comprueba que el problema es complejo computacionalmente. La demostración formal de que el problema de ruteo de vehículos con flexibilidad en la fecha de entrega es NP–completo se presenta en la sección 3.

Para finalizar este capítulo, en la sección 4, se investiga la relación entre el problema que se estudia y el modelo que proponemos con otros modelos afines encontrados en la literatura.

3.1 Modelo I

El primer modelo que se presenta está basado en el del VRP Capacitado (CVRP), descrito a detalle en la sección 2.1.1. El CVRP, considerado como el modelo base de muchos de los problemas de ruteo, consiste en diseñar las rutas óptimas de tal forma que se minimice la distancia total recorrida satisfaciendo todas las demandas de los centros de distribución. Tomando este modelo estándar, para abordar nuestro problema faltaría la modelación de las fechas en que se entregan los productos y la flexibilidad para estas fechas de entrega. Le agregaremos estas condiciones al modelo base.

Para la formulación matemática se definirá un grafo completo. A cada orden o pedido se le asignará un nodo del grafo, al igual que al depósito central. La matriz de distancias (c_{ij}) estará formada por la distancia entre la localización de la orden i y la localización de la orden j. Identificamos entonces cada localización con su orden correspondiente para simplificar notación. Consideraremos

HT = Número de días en el horizonte de planeación

n = número total de órdenes

V = $\{0, 1, 2, \ldots, n\}$ conjunto de órdenes, 0 corresponde al depósito

K = número máximo de vehículos disponibles en un día del horizonte de planeación

R = número máximo de rutas disponibles en el horizonte de planeación

Q = capacidad de los vehíıculos

c_{ij} = distancia entre la localización de la orden i y la localización de la $orden$j, $\forall i, j \in V$

q_i = demanda de la orden i, $\forall i = 1, \ldots, n$

e_i = fecha en que se requiere la entrega de la orden i. En adelante también nos referiremos a ella como *deadline* de la orden i, $\forall i = 1, \ldots, n,\ e_i \in \{1, 2, \ldots, HT\}$

g = número máximo de días que una orden puede ser servida antes de su *deadline*

Para formular el problema con un modelo lineal entero–mixto tenemos las siguientes variables de decisión

$$x_{ij} = \begin{cases} 1, & \text{si el nodo } j \text{ se visita justo después del nodo } i \\ 0, & \text{en otro caso} \end{cases}$$

El modelo para el CVRP, enunciado en 2.1.1 es el siguiente [87]:

$$\text{mín} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

sujeto a

$$\sum_{i \in V} x_{ij} = 1, \ \ \forall j \in V \backslash \{0\}$$
$$\sum_{j \in V} x_{ij} = 1, \ \ \forall i \in V \backslash \{0\}$$
$$\sum_{i \in V} x_{i0} \leq K$$
$$\sum_{j \in V} x_{0j} = \sum_{i \in V} x_{i0}$$
$$u_i - u_j + Q x_{ij} \leq Q - q_j, \quad \forall i, j \in V \backslash \{0\}, \ i \neq j, \ \text{tal que } q_i + q_j \leq Q$$
$$q_i \leq u_i \leq Q, \ \ \forall i \in V \backslash \{0\}$$
$$x_{ij} \in \{0, 1\}, \ \forall i, j \in V$$

Las variables u_i son variables auxiliares que se introducen para establecer la familia de restricciones alternativas equivalentes a las restricciones de capacidad y corte, las cuales tienen cardinalidad polinomial y evitan la formación de subtours. Las variables u_i representan la carga de los vehículos después de visitar el nodo i [87].

Basándonos en este modelo, faltaría agregar las variables y restricciones asociadas a la fecha de entrega y la flexibilidad que se propone en esta fecha de entrega. Tenemos que garantizar que las fechas de requerimiento de productos de todos los nodos que pertenecen a la misma ruta no difieran en más de g días. Para ello impondremos las siguientes restricciones: Si r y s son nodos en la misma ruta, entonces

$$|e_r - e_s| \leq g$$

En este caso diremos que los nodos r y s tienen *deadlines* compatibles.

Para modelar estas restricciones de forma que el modelo permanezca lineal, introduciremos algunas variables auxiliares. Primero consideremos una variable z_j para cada nodo j, que representa una cota inferior de las fechas de requerimiento de productos de los nodos de la ruta a la que pertenezca el nodo j. Esta variable estará acotada entre 1 y el *deadline* del nodo j, esto es,

$$1 \leq z_j \leq e_j$$

Para garantizar que z_j represente la cota inferior de las fechas de requerimiento de la ruta, debe satisfacer que si un vehículo viaja del nodo i al nodo j, la cota inferior del nodo al que llega, debe ser menor o igual a la cota inferior del nodo del cual partió. Esto es, si $x_{ij} = 1$, entonces

$$z_j \leq z_i$$

Esto puede modelarse como

$$z_j - z_i \leq M(1 - x_{ij})$$

para cada $i, j \in V\backslash\{0\}$, con $M > 0$ una constante.

Por otra parte, sea w_j una cota superior de las fechas de requerimientos de productos de los nodos de la ruta a la que pertenece el nodo j. Si $x_{ij} = 1$ necesariamente se debe cumplir

$$w_j \geq w_i$$

ya que la cota superior del nodo de llegada debe ser mayor o igual que la cota superior del nodo de partida y agregamos las restricciones

$$w_i - w_j \leq M(1 - x_{ij})$$

para cada $i, j \in V\backslash\{0\}$.

Para garantizar que las fechas de requerimientos de productos de los nodos que pertenezcan a una ruta no difieran en más de g, agregamos las restricciones

$$w_j - z_j \leq g$$

para cada nodo j.

El modelo entero–mixto propuesto para el VRP con flexibilidad en las fechas de entrega es el siguiente.

$$\text{mín} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

sujeto a

$$\sum_{i \in V} x_{ij} = 1, \qquad \forall j \in V\backslash\{0\} \tag{3.1}$$

$$\sum_{j \in V} x_{ij} = 1, \qquad \forall i \in V \backslash \{0\} \tag{3.2}$$

$$\sum_{i \in V} x_{i0} \leq R \tag{3.3}$$

$$\sum_{i \in V} x_{i0} = \sum_{j \in V} x_{0j} \tag{3.4}$$

$$u_i - u_j + Q x_{ij} \leq Q - q_j, \quad \forall i, j \in V \backslash \{0\}, i \neq j, \tag{3.5}$$

$$\text{tal que } q_i + q_j \leq Q$$

$$q_i \leq u_i \leq Q, \qquad \forall i \in V \backslash \{0\} \tag{3.6}$$

$$z_j - z_i \leq M(1 - x_{ij}), \qquad \forall i, j \in V \backslash \{0\} \tag{3.7}$$

$$w_i - w_j \leq M(1 - x_{ij}), \qquad \forall i, j \in V \backslash \{0\} \tag{3.8}$$

$$w_j - z_j \leq g, \qquad \forall j \in V \backslash \{0\} \tag{3.9}$$

$$1 \leq z_j \leq e_j, \qquad \forall j \in V \backslash \{0\} \tag{3.10}$$

$$e_j \leq w_j \leq HT, \qquad \forall j \in V \backslash \{0\} \tag{3.11}$$

$$u_j, z_j, w_j \geq 0, \qquad \forall j \in V \backslash \{0\} \tag{3.12}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in V \tag{3.13}$$

Las restricciones (3.1) y (3.2) son las restricciones de grado, exactamente un arco llega y sale de cada centtro de distribución. Análogamente, (3.3) y (3.4) imponen las restricciones de grado para el depósito y (3.7) a (3.9) son las restricciones adicionales que se han introducido para modelar la flexibilidad en la fecha de entrega. (3.10) y (3.11) son las restricciones impuestas para las cotas inferior y superior.

Note que con este modelo se obtienen todas las rutas en el horizonte de planeación, pero no garantiza que el número de rutas en un día dado no exceda el número de vehículos disponibles.

Con estas restricciones conseguimos que los vértices (órdenes) que pertenecen a una ruta, difieran en sus *deadlines* a lo más en g. Demostramos enseguida este resultado.

Proposición. En el modelo anterior, si dos vértices r y s pertenecen a la misma

ruta, entonces sus *deadlines* son compatibles, es decir,

$$|e_r - e_s| \leq g$$

Demostración. Supongamos que tenemos una ruta de la forma

$$0, p, r, l, \cdots, q, s, t, 0$$

obtenida con el modelo, entonces se satisface que

$$x_{pr} = x_{rl} = \cdots = x_{qs} = x_{st} = 1$$

de donde

$$z_t \leq z_s \leq \cdots \leq z_r \leq z_p$$

Análogamente, para las w, tenemos

$$w_p \leq w_r \leq \cdots \leq w_s \leq w_t$$

Las dos ecuaciones anteriores se satisfacen para los vértices intermedios de la ruta, y tenemos que todos los elementos cumplen la restricción

$$z_i \leq e_i \leq w_i$$

Entonces se satisface

$$z_t \leq z_s \leq \cdots \leq z_r \leq e_r \leq w_r \leq \cdots \leq w_s \leq w_t$$

de igual forma

$$z_t \leq z_s \leq e_s \leq w_s \leq w_t$$

Utilizando el hecho de que $|w_t - z_t| \leq g$, por la restricción (3.9) se concluye que

$$|e_r - e_s| \leq g$$

lo cual completa la demostración.

□

En total en este modelo se tienen n^2 variables binarias y $3n$ variables continuas, así como $3n^2+4n+2$ restricciones. Se han agregado $2n$ variables continuas y $2n^2+n$ restricciones al modelo VRP Capacitado donde n es el número total de órdenes en el período de planeación.

3.1.1 Validación del modelo

El modelo propuesto fue validado resolviendo instancias de forma exacta utilizando software comercial. Para esto se realizaron algunos experimentos computacionales generando instancias de forma aleatoria y resolviéndolas utilizando Cplex v.11.2.

El programa generador de instancias genera de forma aleatoria los costos, demandas y *deadlines* asociados a cada nodo. Para estas instancias consideramos un horizonte de planeación de una semana, esto es, las fechas de entrega son de lunes a viernes, $HT = 5$, $e_i \in \{1, 2, \ldots, 5\}$. Para la obtención de la matriz de costos se generan n puntos de forma aleatoria en el plano euclideano, en el intervalo $[-25, 25] \times [-25, 25]$, el costo entre el nodo i y el nodo j, c_{ij}, está dado por la función techo de la distancia euclideana entre los dos puntos i y j. De esta forma se garantiza que se satisface la desigualdad triangular, que debe cumplir la matriz de costos para que funcionen la mayoría de los algoritmos para el VRP [87].

El programa genera de forma aleatoria los *deadlines* y las demandas asociadas a cada nodo. Las demandas son generadas en el intervalo $[1, 15]$. Todos los vehículos tienen una capacidad fija de 30. Se proporciona R el número máximo de rutas en todo el horizonte de planeación. Este valor está relacionado con el número de vehículos que se pueden utilizar en cada día del horizonte de planeación: si consideramos la planeación en una semana, de lunes a viernes, entonces $R = 5K$ donde K es el número de vehículos disponibles cada día. Se varía el valor de g, la cota máxima permitida para adelantar días de entrega, esto para ver qué efecto tiene aumentar o disminuir este parámetro.

Se presentan enseguida los resultados de estos experimentos preliminares, los cuales se obtuvieron resolviendo el modelo con Cplex v.11.2 en un servidor Sun Fire V440 con 4 procesadores Ultra Sparc III a 1062 GHZ con 8 Gb de RAM. Cplex se ejecutó con los parámetros por default, en particular con una tolerancia de error relativo de 10^{-4} para el optimizador de problemas entero mixtos. Se reportan el tiempo en segundos que tarda Cplex en resolverlo, el valor de la función objetivo en la solución óptima y el número de iteraciones.

Tabla 3.1: Resultados Modelo I para instancias aleatorias

n	g	Tiempo	Fun. Obj.	Iteraciones
	0	0.03	553	83
	1	3.39	413	21300
	2	5.69	365	37059
10	3	24.8	344	168212
	4	64.11	298	429669
	∞	60.41	298	429669
	0	0.11	590	685
	1	12.11	445	95609
	2	322.84	423	2160604
12	3	1539	399	8553389
	4	4811	376	2512026
	∞	4770	376	25120226
	0	0.31	688	2645
	1	342.27	538	2171299
	2	11385.52	499	53153346
15	3	119285.59	488	4665936567
	4	171280.23	458	617229211
	∞	172686.97	458	617229211

Se observa que conforme aumenta el número de órdenes (n), también aumenta

considerablemente el tiempo necesitado para encontrar la solución, de la misma forma que aumenta el número de iteraciones para encontrar la solución óptima.

Otra observación importante que se deriva es que el número de días permitidos para adelantar pedidos, (g), también influye en el tiempo necesitado por el software para llegar a la solución óptima. Al tener más días permitidos para el adelanto en la fecha de entrega, el espacio de soluciones factibles aumenta y se necesita más tiempo para explorar este espacio y determinar la solución óptima.

Se tiene, como era de esperarse, que para una n fija y variando el parámetro g disminuye el valor de la función objetivo, ya que $g = 0$ corresponde a un ruteo día a día, esto es, sin flexibilidad en la fecha de entrega y $g = 4$ ó ∞ en estos ejemplos, corresponde a un problema de ruteo donde todos los pedidos pueden entregarse desde el primer día del horizonte de planeación, disminuyendo así la distancia total recorrida.

Considerando g fija y variando n, vemos de estos resultados que el tiempo necesario para encontrar la solución óptima crece de forma exponencial al aumentar n. Esto nos muestra empíricamente que el problema es NP–completo, la demostración formal de este hecho la veremos más adelante.

3.2 Modelo II

Una de las limitantes del modelo presentado en la sección anterior es que respeta el número máximo del total de rutas permitidas en el horizonte de planeación sin controlar el número de vehículos (rutas) en cada día. Esto puede se problemático en algunas situaciones, sobre todo en aplicaciones donde el número de vehículos sea muy limitado. Por ello se ha desarrollado un segundo modelo que elimina esta deficiencia. Este segundo modelo puede verse como un modelo intermedio entre el VRP Capacitado y el VRP Periódico, específicamente puede considerarse una generalización del VRP Capacitado y un caso particular del VRP Periódico.

Consideremos la misma notación utilizada para el modelo I, las variables de decisión en este segundo modelo, son las siguientes.

$$x_{ij} = \begin{cases} 1, & \text{si la localización } j \text{ se visita justo después de la localización } i \\ 0, & \text{en otro caso} \end{cases}$$

$\forall i, j \in V$, estas variables se conservan igual que en el modelo I

$$y_{id} = \begin{cases} 1, & \text{si la localización } i \text{ se visita el día } d \\ 0, & \text{en otro caso} \end{cases}$$

$\forall i \in V \backslash \{0\}, \forall d = 1, \ldots, HT$, con estas variables se controla el día de la fecha de entrega en cada localización.

$$z_{id} = \begin{cases} 1, & \text{si la localización } i \text{ se visita el día } d \text{ y es la primera de su ruta} \\ 0, & \text{en otro caso} \end{cases}$$

$\forall i \in V \backslash \{0\}, \forall d = 1, \ldots, HT$, estas variables se introducen para controlar el número de vehículos utilizados cada día.

En este caso, para garantizar que una orden sea servida sólo un día, se tiene para cada nodo i la siguiente restricción

$$\sum_{d=1}^{HT} y_{id} = 1$$

Si en la solución se utiliza el arco (i, j), esto es, si $x_{ij} = 1$, entonces el día de entrega para los nodos i y j debe ser el mismo. Esto se puede lograr imponiendo la restricción

$$y_{id} - y_{jd} \leq 1 - x_{ij}$$

para todos los nodos del grafo.

Las variables $z'_{id}s$ controlarán el número de vehículos utilizados cada día, z_{id} será uno si el nodo i es el primero en la ruta en el día d, esto es, si se satisface que $x_{0i} = 1$ y $y_{id} = 1$. Para lograr esto se impone la siguiente condición para todos los nodos del grafo

$$z_{id} \geq y_{id} + x_{0i} - 1$$

y para no sobrepasar el número de vehículos disponibles tendremos que el total de rutas por día debe ser menor igual a K, esto es,

$$\sum_{i=1}^{n} z_{id} \leq K$$

para toda $d \in \{1, 1, \ldots, HT\}$.

Para controlar las fechas de entrega, se introduce un parámetro auxiliar, denotado por mt_{id}, el cual determinará los días factibles para la entrega de productos en cada nodo. Se define como

$$mt_{id} = \begin{cases} 1, & \text{si la orden } i \text{ puede ser servida en el día } d \\ 0, & \text{en otro caso} \end{cases} \tag{3.14}$$

Por ejemplo, consideremos un horizonte de planeación de 5 días, de lunes a viernes, y supongamos que el nodo r solicita los productos el miércoles, esto es, $e_r = 3$ y que $g = 1$, lo que equivale a permitir sólo un día como máximo el adelanto en la fecha de entrega. Entonces

$$mt_{r2} = mt_{r3} = 1, \qquad mt_{r1} = mt_{r4} = mt_{r5} = 0$$

Para que la fecha de entrega en el nodo i sea factible, se impone la condición

$$y_{id} \leq mt_{id}$$

en cada nodo del grafo.

Con estas condiciones, establecemos enseguida el nuevo modelo, el cual, al igual que el modelo I, toma como base el modelo del VRP Capacitado y se agregan las restricciones para el control de la fecha de entrega y del número de vehículos disponibles para cada día.

El modelo propuesto es el siguiente

$$\text{mín} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

sujeto a

$$\sum_{i \in V} x_{ij} = 1, \qquad \forall j \in V \backslash \{0\} \tag{3.15}$$

$$\sum_{j \in V} x_{ij} = 1, \qquad \forall i \in V \backslash \{0\} \tag{3.16}$$

$$\sum_{i \in V} x_{i0} = \sum_{j \in V} x_{0j} \tag{3.17}$$

$$u_i - u_j + Q x_{ij} \leq Q - q_j, \qquad \forall i, j \in V \backslash \{0\}, i \neq j, \text{ tal que } q_i + q_j \leq Q \tag{3.18}$$

$$q_i \leq u_i \leq Q, \qquad \forall i \in V \backslash \{0\} \tag{3.19}$$

$$y_{id} \leq (mt)_{id}, \qquad \forall i \in V \backslash \{0\}, \forall d \in \{1, 2, \ldots, HT\} \tag{3.20}$$

$$\sum_{d=1}^{HT} y_{id} = 1, \qquad \forall i \in V \backslash \{0\} \tag{3.21}$$

$$y_{id} - y_{jd} \leq 1 - x_{ij}, \qquad \forall i, j \in V \backslash \{0\}, i \neq j \tag{3.22}$$

$$z_{id} \geq y_{id} + x_{0i} - 1, \qquad \forall i \in V \backslash \{0\}, \forall d \in \{1, 2, \ldots, HT\} \tag{3.23}$$

$$\sum_{i=1}^{n} z_{id} \leq K, \qquad \forall d \in \{1, 2, \ldots, HT\} \tag{3.24}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in V \tag{3.25}$$

$$y_{id} \in \{0, 1\}, \qquad \forall i \in V \backslash \{0\}, \forall d \in \{1, 2, \ldots, HT\} \tag{3.26}$$

$$z_{id} \geq 0, \qquad \forall i \in V \backslash \{0\}, \forall d \in \{1, 2, \ldots, HT\} \tag{3.27}$$

Las restricciones (3.15) y (3.16) son las restricciones de grado, exactamente un arco llega y sale de cada centro de distribución. Análogamente, (3.17) impone los requerimientos de grado del depósito; (3.18) y (3.19) son la familia de restricciones alternativas equivalentes a las restricciones de capacidad y corte; (3.20) y (3.21) imponen que cada orden debe ser servida en exactamente un día y que este día pertenece al conjunto de sus días factibles; (3.22) garantiza que las órdenes que pertenezcan a la misma ruta sean servidas el mismo día; (3.23) y (3.24) imponen el número máximo de rutas por día, dependiendo del tamaño de la flotilla de vehículos disponibles.

Las restricciones (3.15) a (3.19) son utilizadas muy frecuentemente en las

formulaciones para el VRP Capacitado. Las restricciones (3.20) a (3.24) han sido diseñadas especificamente para modelar la flexibilidad en la fecha de entrega.

3.2.1 Validación del modelo

El modelo fue validado, al igual que el modelo I, en instancias de prueba que se resolvieron con Cplex v. 11.2, el cual se ejecutó con los parámetros por default, en particular con una tolerancia de error relativo de 10^{-4} para el optimizador de problemas entero mixtos. Se utilizaron los mismos datos de las instancias descritas en la sección 3.1.1 y los resultados obtenidos se presentan en la tabla 3.2.

Se observa que se obtienen resultados similares a los obtenidos en el Modelo I sobre estas mismas instancias de prueba. El tiempo crece al aumentar n y la distancia disminuye al considerar varios días de flexibilidad en la fecha de entrega. La solución de las instancias con el Modelo II consume, en general, más tiempo que con el Modelo I.

Cabe hacer notar que los valores de la función objetivo, son los mismos, salvo cuatro resultados de la tabla anterior. En el caso $n = 10$, $g = 4, \infty$ en el Modelo I se obtiene el valor de 298 en el punto óptimo y de 300 en el Modelo II. Recordemos que uno de los inconvenientes del Modelo I es que controla el número total de rutas en todo el horizonte de planeación, pero no el número de rutas de cada día. En este caso la solución con valor de 298 requiere más rutas de las permitidas por día, pero no sobrepasa el total por semana, por eso se obtiene un menor valor de la función objetivo en el óptimo en una solución que no es factible para el Modelo II. Lo mismo ocurre para $n = 12$ y $g = 4, \infty$ que con el Modelo I obtiene un menor valor de la función objetivo que con el Modelo II.

En situaciones donde el número de vehículos sea muy limitado es aconsejable utilizar el Modelo II, de otra forma el Modelo I es adecuado.

En ambos casos, se tiene que los modelos son complejos computacionalmente, se observa que aumenta considerablemente el tiempo utilizado para llegar al punto

óptimo al aumentar el número de órdenes. En la siguiente sección se aborda este tema más formalmente.

Tabla 3.2: Resultados Modelo II para instancias aleatorias

n	g	Tiempo	Fn. Obj.	Iteraciones
	0	0.03	553	84
	1	3.42	413	19717
	2	8.99	365	55815
10	3	54.95	344	325951
	4	79.49	300	500956
	∞	76.5	300	500956
	0	0.10	590	685
	1	11.11	445	84053
	2	160.52	423	1075942
12	3	1658.00	399	8568862
	4	10295.00	393	53309717
	∞	10241.00	393	53309717
	0	0.36	688	2455
	1	284.57	538	1770057
	2	53278.20	499	242719367
15	3	110621.63	488	576284540
	4	615973.15	458	1857912021
	∞	615960.26	458	1857912021

3.3 Complejidad

En esta sección utilizaremos algunos resultados de la teoría de Complejidad Computacional que pueden consultarse en [77]. Mostraremos que el problema de ruteo de vehículos con flexibilidad en las fechas de entrega es NP–completo.

Primero repasaremos la demostración de que el VRP Capacitado (CVRP), es NP–completo [65] y en base a esto, se mostrará que el problema con flexibilidad en las fechas de entrega también es NP–completo.

De la definición formal tenemos que **un problema es NP–completo si pertenece a la clase NP y se puede reducir a algún problema NP–completo conocido**.

Veremos enseguida qué significa que un problema pertenezca a la clase NP y el concepto de reducción.

Para la demostración necesitamos trabajar con la versión de decisión del problema. Un problema es de decisión si su respuesta es "sí"ó "no". Cada problema de optimización tiene asociado su correspondiente problema de decisión, los resultados obtenidos sobre este problema de decisión son aplicables al problema de optimización.

La versión de decisión del CVRP es la siguiente: consideremos G un grafo completo con n nodos y K vehículos con capacidad Q. El problema de decisión asociado es: Dado un presupuesto L ¿Existen K ciclos que visiten todos los nodos satisfaciendo las demandas de los clientes y capacidad de los vehículos con un costo menor o igual a L? En este caso solo se puede tener la respuesta sí o no.

Decimos que un problema pertence a la clase NP si podemos codificar una solución al problema y verificar la respuesta "si" en tiempo polinomial.

Un concepto muy importante en la teoría de complejidad es el concepto de reducción. Decimos que el problema A es por lo menos tan difícil como el problema B si B se reduce a A. Decimos que B reduce a A si existe una transformación R

la cual, para cada entrada x de B produce una entrada equivalente $R(x)$ de A. Por equivalente, se entiende que la respuesta a $R(x)$ considerada como entrada de A, si o no, es la respuesta correcta a x considerada como entrada de B, esto es,

$$x \in B \Longleftrightarrow R(x) \in A \tag{3.28}$$

En otras palabras, para resolver B en la entrada x tenemos que calcular $R(x)$ y resolver A en esta entrada, esto se ilustra en la figura 3.1.

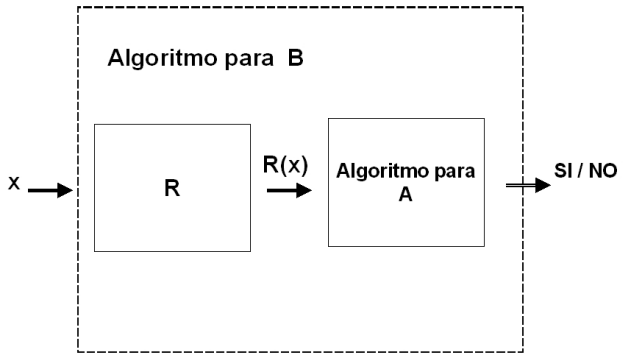


Figura 3.1: Reducción de B en A

El cálculo de $R(x)$ no debe ser complicado, asumiremos que una reducción es eficiente si ésta se logra en espacio logarítmico, esto es, el espacio necesitado para construir $R(x)$ está acotado por una función logarítmica.

Para lograr la reducción necesitamos tres pasos: construir $R(x)$, demostrar (3.28) y demostrar que R se construye en espacio logarítmico.

Con estos conceptos, tenemos que para demostrar que un problema P es NP–completo se necesitan dos condiciones

1. P pertenece a la clase NP
2. P se puede reducir a algún problema NP–completo conocido

3.3.1 El problema de ruteo de vehículos con flexibilidad en las fechas de entrega es NP–completo

Veremos primero que el CVRP es NP–completo. Tenemos que el CVRP pertenece a la clase NP ya que dada una solución, que consta de K ciclos donde se satisfacen las demandas de todos los clientes y no se sobrepasa la capacidad de los vehículos, se puede verificar en tiempo polinomial si el costo de la solución propuesta es menor que L, esto porque lo único que se tiene que hacer es calcular los costos de cada ruta y sumarlos para obtener el costo total. Esto se puede hacer en tiempo polinomial. Por lo tanto se satisface 1), CVRP $\in$ NP.

Para mostrar que CVRP también satisface 2) mostraremos una reducción del problema del agente viajero, TSP, al CVRP, y como TSP es NP–completo [77], entonces habremos mostrado 2) y por lo tanto que CVRP también es NP– completo.

Reducción de TSP a CVRP

Para ver esta reducción, tenemos que construir la relación R, ver que satisface (3.28) y mostrar que se construye en espacio logarítmico.

Construcción de la relación R. Supongamos $G = (V, A)$ un grafo completo que sea una instancia del TSP.

La relación R tiene como entrada el grafo G y como salida considera el mismo grafo y $K = 1$, esto es, un solo vehículo con demandas 1 en cada nodo, ésta es una instancia del CVRP. Con esto tenemos la construcción.

Equivalencia. $\Longrightarrow$) Consideremos la instancia del TSP con un presupuesto menor que L, entonces por la forma que está construida la relación la instancia asociada del VRP tendrá un solo vehículo que satisface las demandas de los clientes y tendrá un presupuesto menor que L, entonces si se satisface la instancia TSP se satisface la instancia CVRP.

$\Longleftarrow$) Consideremos una instancia $R(x)$, esto es, una instancia del CVRP con un solo vehículo con capacidad suficiente para todos los clientes con presupuesto L, tenemos que claramente esta instancia satisface TSP con el mismo presupuesto L.

Construcción en espacio logarítmico. La construcción se hace en espacio logarítmico ya que trabajamos con el mismo grafo, solo generamos la variable K con valor 1 y la capacidad del vehículo que es la suma de las demandas de los nodos, todo esto se logra en espacio logarítmico.

Tenemos completa la reducción de TSP a CVRP, por lo tanto concluimos que el CVRP es NP–completo.

Veremos ahora que el VRP con flexibilidad en las fechas de entrega también es NP–completo. La versión de decisión del problema es: Dado un presupuesto L ¿Existen K ciclos que visiten todos los nodos, satisfaciendo las demandas de los clientes y capacidad de los vehículos, permitiendo adelantar a lo más g días la fecha de entrega, de con un costo menor o igual a L?

Dada una solución que satisface todas las restricciones, se puede verificar en tiempo polinomial si el costo es menor igual a L, por lo tanto el VRP con flexibilidad en fechas de entrega $\in$ NP. Veremos enseguida la reducción al CVRP para completar la demostración que nuestro problema es NP–completo.

Reducción de CVRP a VRP con flexibilidad en fechas de entrega

Construiremos la relación R que satisface (3.28) y mostraremos que se construye en espacio logarítmico.

Construcción de la relación R. Consideremos una instancia del CVRP formada por un grafo G, demandas asociadas en cada nodo y K vehículos disponibles.

La relación R tendrá como entrada esta instancia del CVRP y como salida considera la misma instancia con $g = \infty$, esto es como no tener restricción con la fecha de entrega, y con *deadlines* 1 en cada nodo. Esta es una instancia donde se puede considerar flexibilidad en las fechas de entrega.

Equivalencia. $\Longrightarrow$) Partiendo de la instancia del CVRP con un presupuesto menor que L, este mismo presupuesto será suficiente para el VRP con flexibilidad en las fechas de entrega, si se consideran todas las entregas el mismo día. Entonces si se satisface la instancia CVRP se satisface la instancia VRP con flexibilidad en las fechas de entrega.

$\Longleftarrow$) Consideremos ahora una instancia $R(x)$, esto es, una instancia del VRP con flexibilidad en las fechas de entrega y $g = \infty$ con presupuesto L, también el CVRP se satisface con este presupuesto, ya que se pueden generar las mismas rutas, todas con el mismo día de entrega.

Construcción en espacio logarítmico. La construcción se hace en espacio logarítmico ya que trabajamos con el mismo grafo, demandas y número de vehículos. Sólo se generan el vector de deadlines de tamaño n y la variable g con valor muy grande, todo esto se logra en espacio logarítmico.

Tenemos completa la reducción de CVRP a VRP con flexibilidad en las fechas de entrega, por lo tanto concluimos que nuestro problema es NP–completo.

Las implicaciones de eso nos llevarán a buscar métodos alternativos para buscar la solución óptima, ya que con esto se muestra que los métodos exactos no son adecuados para tratar problemas prácticos donde usualmente se tienen instancias con un gran número de nodos.

3.4 Relación con otros modelos de la literatura

El problema que estamos trabajando tiene relación con algunos modelos reportados en la literatura. En particular, veremos la relación del VRP con flexibilidad en las fechas de entrega y las formulaciones propuestas en las secciones 3.1 y 3.2 con el problema de ruteo de vehículos con ventanas de tiempo (VRP-TW) y el VRP periódico (PVRP). Tanto el VRP-TW como el PVRP fueron introducidos en la sección 2.1.2.

3.4.1 Relación con el problema de ruteo de vehículos con ventanas de tiempo

Estudiaremos la relación del problema de ruteo de vehículos con flexibilidad en las fechas de entrega aqui abordado y el VRP con ventanas de tiempo, el cual puede describirse como un VRP Capacitado, (CVRP), en donde cada cliente tiene que ser visitado una sola vez por un vehículo en un intervalo de tiempo dado. Además de las condiciones impuestas en el CVRP se tiene que en el VRP-TW, adicionalmente al costo de transporte o distancia c_{ij} entre el nodo i y el nodo j, se maneja un tiempo de viaje t_{ij} asociado a cada arco $(i, j) \in A$ y un tiempo de servicio s_i para cada nodo i. Usualmente en las formulaciones asociadas se duplica el nodo del depósito, agregando el nodo $n + 1$ al grafo, las rutas se consideran que parten del nodo 0 y llegan al nodo $n + 1$.

Cada cliente i tiene asociado un intervalo de tiempo $[a_i, b_i]$ en donde puede recibir el servicio. Los vehículos deben salir del depósito en el tiempo $[a_0, b_0]$ y regresar en el tiempo $[a_{n+1}, b_{n+1}]$. Un vehículo puede llegar al nodo i antes del tiempo a_i, pero tiene que esperar para entregar los productos, esto no genera algún costo, pero aumenta el tiempo total de recorrido. Sin embargo, no se permite que un vehículo llegue al nodo i después del tiempo b_i.

En el caso de nuestro problema de VRP con flexibilidad en las fechas de entrega, podemos considerar la flexibilidad en la entrega como los intervalos permitidos para entregar los productos. Esto es, si el nodo i tiene fecha de entrega e_i y se permite una flexibilidad de g días para adelantar pedidos, entonces tendremos que la ventana de tiempo para el nodo i estará dada por

$$[a_i, b_i] = [e_i - g, e_i]$$

Siempre y cuando $e_i - g \geq 1$ de otra forma tendremos el inicio de la ventana el primer día.

Por ejemplo, supongamos que tenemos los siguientes cinco clientes con sus respectivas fechas de entrega

cliente	*deadline*
A	2
B	5
C	1
D	4
E	3

Si consideramos $g = 1$, esto es, permitir el adelanto en la fecha de entrega de un día, tenemos que las ventanas de tiempo asociadas a cada nodo son

$$\begin{aligned} [a_A, b_A] &= [1, 2] \\ [a_B, b_B] &= [4, 5] \\ [a_C, b_C] &= [1, 1] \\ [a_D, b_D] &= [3, 4] \\ [a_E, b_E] &= [2, 3] \end{aligned}$$

La formulación del VRP-TW considera las siguientes variables de decisión

$$x_{ij}^k = \begin{cases} 1 & \text{si el vehículo } k \text{ viaja del nodo } i \text{ al nodo } j \\ 0 & \text{en otro caso} \end{cases}$$

$$s_i^k = \text{tiempo en que el vehículo } k \text{ inicia el servicio con el cliente } i$$

Se asume que $s_0^k = 0 \ \forall k$ y s_{n+1}^k es el tiempo en que llega el vehículo k al depósito.

Una de las desventajas de formular el problema con flexibilidad en las fechas de entrega como un VRP-TW es que este modelo no contempla la restricción de que los vehículos partan y retornen al depósito el mismo día. Por consiguiente, muchas de las soluciones que serían factibles en el VRP-TW, serían infactibles para nuestro problema. Otro inconveniente es que se tendría que especificar para cada nodo y cada arco del grafo el tiempo de servicio s_i y el tiempo de transportación t_{ij}. En el ejemplo éstos tienen que estar dados en fracciones de días. Hay algunas aplicaciones, como la que nos ocupa en esta tesis, donde el objetivo prioritario es minimizar la distancia y el tiempo no representa inconvenientes, cuando las entregas pueden realizarse en un solo día sin problema.

Otra desventaja es que se tiene una situación similar al modelo I, presentado en la sección 3.1, ya que se trabaja sobre el total de rutas en todo el horizonte de planeación, pero no se controla el número de rutas en cada día.

En total el VRP-TW maneja Rn^2 variables y Rn variables continuas, donde R es el número máximo de rutas en todo el horizonte de planeación. El modelo I maneja n^2 variables binarias y $3n$ continuas.

3.4.2 Relación con el problema de ruteo de vehículos periódico

Estudiaremos ahora la relación del VRP con flexibilidad en las fechas de entrega y el VRP Periódico (PVRP), el cual busca diseñar las rutas para satisfacer las demandas de un conjunto de clientes sobre múltiples días. Se cuenta con un horizonte de planeación D y se definen calendarios para cada cliente, esto es, un calendario es un colección de días en el horizonte de planeación en los cuales los clientes recibirán el servicio. Asignar un nodo a un calendario implica que el nodo recibirá el servicio en cada día del calendario.

Tenemos

$$S = \text{ Conjunto de calendarios}$$

$$s \in S \text{ un calendario}$$

Cada calendario en S está descrito por un vector a_{sd} tal que

$$a_{sd} = \begin{cases} 1 & \text{si el día } d \in D \text{ pertenece al calendario } s \in S \\ 0 & \text{en otro caso} \end{cases}$$

Entonces lo que se busca es seleccionar para cada nodo un calendario de un conjunto candidato, asignar nodos a vehículos para cada día y diseñar las rutas para cada día en el horizonte de planeación. El modelo general considera que cada nodo requiere un número de visitas f_i durante el horizonte de planeación y para cada nodo i se debe escoger un calendario de un subconjunto factible de calendarios, esto es, cada nodo i tendrá asociado un conjunto S_i, tal que

$$S_i \subset S, \qquad S_i = \left\{ s \in S : \sum_{d \in D} a_{sd} = f_i \right\}$$

En nuestro caso del VRP con flexibilidad en la fecha de entrega $f_i = 1$, $\forall i = 1, 2, \ldots, n$, esto es, estamos considerando que todos los nodos requieren una sola visita en todo el horizonte de planeación. Además, cada calendario constaría de un solo elemento. Por ejemplo, si consideramos un horizonte de planeación de 5 días, de lunes a viernes, y con los siguientes *deadlines*

cliente	*deadline*
A	2
B	5
C	1
D	4
E	3

Si consideramos $g = 1$, esto es, permitir el adelanto en la fecha de entrega de un día, tenemos que por ejemplo el cliente A tendrá dos posibles días para la entrega

de productos, el día 1 ó el día 2. El cliente B podrá recibir productos el día 4 ó el día 5. El cliente C sólo puede recibir productos el día 1 y así sucesivamente. Entonces el conjunto de calendarios S está dado por

$$S = \{s_1, s_2, s_3, s_4, s_5\}$$

y

$$s_1 = \{1\}, \quad s_2 = \{2\}, \quad s_3 = \{3\}, \quad s_4 = \{4\}, \quad s_5 = \{5\},$$

Cada calendario consta de un solo elemento; hay que escoger para cada cliente de entre sus posibles calendarios uno para definir la fecha de entrega. Por ejemplo, el conjunto de calendarios factibles para el cliente A es

$$S_A = \{s_1, s_2\}$$

para el cliente B es

$$S_B = \{s_4, s_5\}$$

para el cliente C es

$$S_C = \{s_1\}$$

La formulación para el PVRP considera las siguientes variables de decisión

$$x_{ijk}^d = \begin{cases} 1 & \text{si el vehículo } k \text{ utiliza el arco } (i,j) \text{ en el día } d \\ 0 & \text{en otro caso} \end{cases}$$

$$y_{ik}^s = \begin{cases} 1 & \text{si el vehículo } k \text{ visita el nodo } i \text{ en el calendario } s \in S \\ 0 & \text{en otro caso} \end{cases}$$

$$z_i^s = \sum_k y_{ik}^s = \begin{cases} 1 & \text{si el nodo } i \text{ es visitado en el calendario } s \in S \\ 0 & \text{en otro caso} \end{cases}$$

$$v_i^d = \begin{cases} 1 & \text{si el nodo } i \text{ es visitado el día } d \\ 0 & \text{en otro caso} \end{cases}$$

En total este modelo maneja $|D|(Kn^2+n)+|S|(nK+n)$ variables binarias. Como en el caso del VRP con flexibilidad en las fechas de entrega cada calendario está definido

como un día en el horizonte de planeación se tiene que en este caso, $|D| = |S|$, por lo que en total se manejarían $|D|(Kn^2 + nK + 2n)$ variables binarias.

Este modelo sí controla el número de vehículos utilizados en cada día del horizonte de planeación, al igual que lo hace el modelo II propuesto en este trabajo, el cual utiliza en su formulación $n^2 + |D|n$ variables binarias y $|D|n$ variables continuas.

El VRP con flexibilidad en las fechas de entrega puede verse como un caso especial del PVRP donde la frecuencia de las visitas a cada nodo es uno y cada calendario consta de un solo elemento. La formulación del modelo II, propuesto en la sección 3.2, explota estas características del problema para dar lugar a un modelo más simplificado y adecuado al problema que estamos trabajando. Asímismo, las metodologías de solución que aquí se proponen también aprovechan estas características por lo que resultan más eficientes para el problema que nos ocupa.

Capítulo 4

Metodologías de solución para el problema mono-objetivo

En el capítulo 3 se demostró que el problema bajo estudio es NP–completo, lo que indica que sólo podremos obtener resultados exactos para instancias pequeñas en tiempos de cómputo razonables. Para instancias grandes, tardaría demasiado obtener la solución óptima. Esta situación prevalece para la mayoría de los problemas de ruteo que se presentan en la práctica, por lo que recientemente se han desarrollado muchos métodos heurísticos para problemas de ruteo y en últimas fechas más los metaheurísticos, para obtener soluciones aproximadas.

Describiremos en este capítulo dos propuestas de metaheurísticas para resolver nuestro problema con un solo objetivo. La primera basada en un método híbrido que combina GRASP (descrita en 2.2.1) con un encadenamiento de trayectorias (Path Relinking) y la segunda basada en una búsqueda por entornos variables (VNS, descrita en 2.2.1).

El capítulo está dividido en dos secciones, en la primera, 4.1, se presenta la propuesta de GRASP con encadenamiento de trayectorias, en 4.1.1 se describe el método constructivo para la generación de soluciones factibles, las cuales son mejoradas con un proceso de búsqueda local descrito a detalle en 4.1.2. El apartado 4.1.3 está dedicado a la descripción del proceso de encadenamiento de trayectorias. En la sección 4.2 se describirá la propuesta de solución utilizando búsqueda con entornos

variables.

4.1 Una metodología de solución: Híbrido de GRASP y encadenamiento de trayectorias (HGET)

El primer procedimiento de solución que proponemos es un algoritmo de dos fases. La primera fase está basada en la metaheurística GRASP, aquí se genera un conjunto de buenas y diversas soluciones. La segunda fase, que puede verse como un postprocesamiento, está basada en un encadenamiento de trayectorias. En esta segunda fase cada par de buenas soluciones obtenidas en la fase I, se combinan buscando soluciones de mejor calidad.

Como se explicó en el capítulo 2, GRASP [36] o Procedimiento de Búsqueda Miope Aleatorizada Adaptativa (Greedy Randomized Adaptive Search Procedure) es una metaheurística que construye soluciones utilizando una función miope adaptativa que incluye un factor de aleatoriedad controlado. Las soluciones se construyen agregando elementos a una solución parcial. La función miope adaptativa estima el beneficio de incluir un elemento candidato en la solución. Se forma una lista restringida de elementos candidatos, conformada por los elementos con los mejores valores de la función miope; de esta lista es seleccionado un elemento aleatoriamente, el cual es incluído en la solución. Se actualizan los valores de la función miope adaptativa y se repite el proceso hasta completar una solución.

Si este proceso es ejecutado varias veces, se espera que se genere una diversidad de soluciones, lo que permitirá explorar el espacio de búsqueda. A cada una de las soluciones generadas se aplica un proceso de búsqueda local para mejorarlas en términos de la función objetivo.

Un bosquejo de esta fase es la siguiente.

Fase I: GRASP

Repetir

I. 1 Constuir una solución utilizando un **método constructivo miope aleatorizado**

I. 2 Mejorar esta solución por un método de **búsqueda local**

Hasta que se satisfaga el criterio de paro

En este caso el criterio de parada se alcanza después de un número predeterminado de iteraciones. Al término esta fase se forma un conjunto con las mejores soluciones generadas. Definamos este conjunto como $SetofBest$ que será donde se aplicará la fase 2 del algoritmo.

El encadenamiento de trayectorias orignalmente fue propuesto por Glover [45] como una estrategia de intensificación que explora trayectorias que conectan soluciones de buena calidad obtenidas por búsqueda tabú o búsqueda dispersa [47]. La idea principal es que en la trayectoria entre dos buenas soluciones pueden existir soluciones de buena calidad, o incluso mejores soluciones. Iniciando en una o más soluciones elite, se generan trayectorias en el espacio solución para llegar a otra solución elite (denominada guía) y en este trayecto se explora el espacio buscando mejores soluciones. Para generar las trayectorias se seleccionan movimientos que introduzcan en la solución actual atributos que están presentes en la solución élite guía.

El encadenamiento de trayectorias da lugar a la segunda fase del algoritmo, que puede describirse como sigue.

Fase II: Encadenamiento de trayectorias

Para cada par de soluciones S,$S' \in SetofBest$ hacer

II.1 Generar un conjunto de soluciones en la trayectoria entre S y S' usando un **método generador de trayectorias**

II.2 Mejorar estas soluciones empleando un método de **búsqueda local**

Enseguida daremos una descripción detallada de los procedimientos utilizados en estas dos fases del algoritmo propuesto, esto es, del método constructivo miope aleatorizado, la búsqueda local y el método generador de trayectorias que han sido diseñados.

4.1.1 Método constructivo miope aleatorizado

Recuérdese que una solución a nuestro problema consiste en asignar a cada centro de distribución el día en que será servido y luego para cada día de la semana determinar el orden en que serán visitados los centros de distribución (o los clientes) asignados a ese día.

El método constructivo consiste en diseñar K clusters o grupos de nodos, uno para cada vehículo, y posteriormente resolver el Problema del Agente Viajero, TSP, para los nodos que pertenecen a cada cluster. La fase de agrupamiento está basada en la idea de Fisher y Jaikumar de abordar el problema como un problema de asignación generalizada (GAP) [38], luego, para resolver el GAP utilizaremos una modificación del algoritmo de Martello y Toth [66].

El algoritmo de esta fase es el siguiente.

I.1: Método Constructivo Miope Aleatorizado

I.1.1 Construir K grupos de nodos como sigue:

- Aproximar el problema por un problema de asignación generalizada, (GAP).
- Resolver el GAP.

I.1.2 Diseñar la ruta en cada grupo de nodos.

A continuación describimos a detalle los elementos planteados en este método, esto es, las ideas utilizadas del algoritmo de Fisher y Jaikumar y del algoritmo de Martello y Toth para construir K grupos de nodos (I.1.1), así como la heurística GENI para diseñar la ruta en cada grupo de nodos (I.1.2).

Construcción de K grupos de nodos

Para la construcción de los K grupos de nodos nos basaremos en el algoritmo de Fisher y Jaikumar que es un algoritmo bien conocido para construir soluciones al VRP que está dentro de la categoría de agrupar primero, rutear después. Considera el número de rutas fijo y las coordenadas de los nodos en el plano euclideano. Sea K el número de rutas que se desean formar. Primero determina los llamados puntos semilla, K, el mismo número de vehículos disponibles y por consiguiente de rutas a programar. Luego calcula los costos asociados, a_{il}, de asignar cada nodo i al cluster l y transforma el problema en un problema de asignación generalizada. Al resolver este GAP se tienen los grupos de nodos que constituirán cada ruta.

En particular, nos interesa el método para seleccionar los puntos semilla el cual describimos a continuación. Recordemos que el conjunto de vértices está dado por $V = \{v_0, v_1, v_2, \ldots v_n\}$, con v_0 el depósito.

Selección de puntos semilla según el algoritmo de Fisher y Jaikumar

1. Determinar las semirectas que unen el origen con cada punto $v_i \in \{v_1, v_2, \ldots, v_n\}$
2. Hallar las bisectrices de los ángulos que forman cada par de semirectas consecutivas. Cada ángulo γ_i entre dos bisectrices consecutivas corresponde a un punto v_i y lleva asociado un peso d_i correspondiente a la demanda del punto.
3. Comenzando por un punto cualquiera y "barriendo" en el mismo sentido ir formando K ángulos mayores η_j, con los ángulos γ_i ó con partes proporcionales de éstos, de forma tal que la demanda total de estos nuevos ángulos sea
$$P = \sum_{i=1}^{n} \frac{d_i}{K}$$
4. Cada punto semilla s_j se situará en la bisectriz del ángulo η_j de forma que la suma de las demandas de los clientes en η_j que queden por debajo del arco que pasa por s_j trazado desde el origen más una proporción de la demanda del cliente más cercano a este arco por fuera sea $0.75P$.

   Esta proporción es $\frac{A}{A+B}$ donde A y B son las distancias a los puntos mas cercanos al arco por dentro y por fuera, respectivamente.

Nótese que en el punto 3 del algoritmo anterior se puede tomar cualquier punto inicial como punto de partida. Así, si se consideran varios puntos iniciales podremos formar varios puntos semilla que darán lugar a diferentes soluciones.

Como se comentó anteriormente, una vez que se determinan los puntos semilla se plantea un problema de asignación generalizada, para determinar la asignación

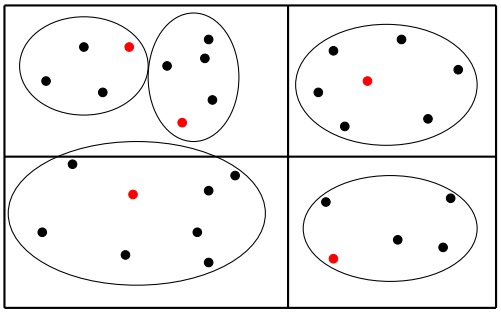


Figura 4.1: Asignación de nodos a puntos semilla

de los nodos o vértices a cada punto semilla y conformar los clusters. Se consideran las variables siguientes:

$$y_{il} = \begin{cases} 1, & \text{si el nodo } i \text{ se asigna al grupo } l \\ 0, & \text{en otro caso} \end{cases}$$

El GAP se puede formular como

$$\text{mín} \sum_{i=1}^{n} \sum_{l=1}^{K} a_{il} y_{il}$$

sujeto a

$$\begin{aligned} &\sum_{i=1}^{n} y_{il} d_i \leq Q, && l = 1, \ldots, K \\ &\sum_{l=1}^{K} y_{il} = 1, && i = 1, \ldots, n \\ &y_{il} = \{0, 1\}, && i = 1, \ldots, n,\ l = 1, \ldots, K \end{aligned}$$

El costo a_{il} es el costo de insertar el nodo i en la ruta que va del depósito al punto semilla s_l.

El GAP es también un problema NP–completo, por lo que necesitaremos un método heurístico para resolverlo. Una de las heurísticas más conocidas para resolver este problema es el algoritmo de Martello y Toth [66], el cual describimos enseguida.

Algoritmo de Martello y Toth para resolver el GAP

1. Hacer $Q(l) = Q$, $l = 1, \ldots, K$, $U = \{1, 2, \ldots, n\}$, $y_{il} = 0$
2. Mientras $U \neq \emptyset$ hacer

   $\forall i \in U:$ determinar $\Delta_{il'} = \text{mín}\{a_{il} : d_i \leq Q(l), l = 1, \ldots K\}$

   determinar $\Delta_{il''} = \text{mín}\{a_{il} : d_i \leq Q(l), l = 1, \ldots K,$
   $l \neq l'(i)\}$

   hacer $b_i = \Delta_{il''} - \Delta_{il'}$
3. determinar $b_{i'} = \text{máx}\{b_i : i \in U\}$
4. Hacer $y_{i'l'} = 1$, $U = U \backslash \{i'\}$, $Q(l') = Q(l') - d(i')$

Como veremos enseguida, tomamos como base este algoritmo para resolver el GAP, pero con algunas variantes para obtener diversidad en las soluciones.

Con los elementos anteriormente explicados estamos en condiciones de describir el algoritmo diseñado para la asignación de nodos a clústers o grupos, esto es, el paso 1.1.1 del método constructivo miope aleatorizado.

Algoritmo propuesto para la asignación de nodos a clusters

Sea a_{il} el costo de insertar el nodo i en la ruta que va del depósito al punto semilla s_l, el cual se obtiene por el algoritmo propuesto por Fisher y Jaikumar. El algoritmo para la asignación de nodos a clusters puede describirse de la siguiente forma.

I.1.1 Algoritmo para la asignación de nodos a clusters

Inicializar $A = \emptyset, P = V \backslash \{0\}$

Repetir

1. Calcular

$$\Delta_i = a_{il^{**}(i)} - a_{il^*(i)}, \text{ para todo } i \in P \backslash A$$

donde

$$\begin{aligned} l^*(i) &= \text{argmin}\{a_{il} \text{ tal que la asignación } i \text{ a } l \text{ sea factible para} \\ &\quad l = 1, 2, \ldots, K\} \\ l^{**}(i) &= \text{argmin}\{a_{il}, l \neq l^*(i), \text{ tal que la asignación } i \text{ a } l \text{ sea factible para} \\ &\quad l = 1, 2, \ldots, K\} \end{aligned}$$

2. Calcular $\Delta_{\text{mín}} = \text{mín}\{\Delta_i | i \in P \backslash A\}$ y $\Delta_{\text{máx}} = \text{máx}\{\Delta_i | i \in P \backslash A\}$
3. Construir $RCL = \{i \in P \backslash A | \Delta_i \geq \alpha \Delta_{\text{mín}} + (1 - \alpha)\Delta_{\text{máx}}\}$
4. Escoger $i^* \in RCL$ aleatoriamente
5. Asignar i^* a $l^*(i^*)$ y hacer $A \cup \{i^*\}$

Hasta que $A = P$

Como se puede observar, en cada iteración no necesariamente se escoge el vértice con el valor más alto de Δ_i como se hace en el algoritmo de Martello y Toth, sino que se construye una lista restringida con los mejores candidatos (RCL), que corresponden a los valores más altos de Δ_i. Luego de esta lista se escoge aleatoriamente un vértice y se asigna a su mejor cluster. El parámetro α controla el nivel de aleatorización, si $\alpha = 0$ entonces la RCL está compuesta sólo por el vértice con el valor más alto de $\Delta_{\text{máx}}$, dando lugar a la versión determínistica. Si $\alpha = 1$, entonces la $RCL = P \backslash A$, esto es, la selección se hace completamente aleatoria. Se debe escoger un valor de α que proporcione un balance entre diversificación y calidad de soluciones.

Para agregarle un factor adicional de diversificación, se ha añadido una modificación a la forma de calcular los costos en el algoritmo propuesto por Fisher y Jaikumar. No consideraremos varios puntos iniciales para el algoritmo de Fisher y

Jaikumar, esto es, se genera solo un conjunto de puntos semilla. Esto implicaría que los coeficientes del GAP, a_{il}, como están definidos hasta el momento, sean únicos, entonces le agregamos el factor de diversidad precisamente a estos costos del GAP. Sea

$$frec(i,l) \;=\; \text{Número de veces que el nodo } i \text{ se ha asignado al cluster } l \text{ en construcciones anteriores, } \forall i \in V\backslash\{0\}$$

$$frec_{\text{máx}(i)} = \text{máx}\{frec(i,l),\ \forall\, l = 1,\ldots,K\},\ \forall i \in V\backslash\{0\}$$

$$a_{\text{máx}_i} = \text{máx}\{a_{il},\ \forall\, l = 1,\ldots,K\},\ \forall i \in V\backslash\{0\}$$

$$a_{\text{mín}_i} = \text{mín}\{a_{il},\ \forall\, l = 1,\ldots,K\},\ \forall i \in V\backslash\{0\}$$

y

$$a'_{il} = a_{il} - \beta\left(\frac{frec(i,l)}{frec_{\text{máx}(i)}}\right)(a_{\text{máx}_i} - a_{\text{mín}_i}),\ \forall l = 1,\ldots,K,\ \forall i \in V\backslash\{0\}$$

En el algoritmo I.1.1 se calculará

$$\Delta_i = a'_{il^{**}(i)} - a'_{il^{*}(i)},\ \text{para todo } i \in P\backslash A$$

Esto es, se utilizan los coeficientes con la penalización. Estas penalizaciones ayudan al método a realizar diferentes asignaciones en cada ejecución del algoritmo. De esta forma se aumenta el nivel de diversificación y se ha observado que con un valor adecuado de β se logra un incremento en la calidad de las soluciones obtenidas.

Una vez que han quedado conformados los grupos de nodos, el diseño de la ruta en cada cluster, (paso I.1.2), se realiza utilizando la heurística de inserción generalizada GENI desarrollada por Gendrau et al. [41] y descrita en el apéndice A. GENI es un método constructivo para el problema del agente viajero, que ha mostrado ser muy efectivo y es por ello que se está utilizando esta heurística para la construcción de cada ruta.

Con todos los elementos descritos anteriormente, el algoritmo para la fase I donde se obtienen soluciones factibles al problema puede enunciarse de la siguiente forma.

Algoritmo Fase I: GRASP

1. Inicializar frecuencias = 0
2. Determinar coeficientes a_{il} con el algoritmo original de Fisher y Jaikumar
3. Repetir
   a) Determinar a'_{il} (que depende de β)
   b) Aplicar el algoritmo I.1.1 para la asignación de nodos a clusters con los valores de a'_{il}
   c) Actualizar frecuencias
   d) Determinar rutas con GENI
   e) Aplicar Búsqueda local

   Hasta satisfacer el criterio de paro

Conviene aclarar que en este algoritmo con cada ejecución del paso 3 completo se obtiene una solución al problema, por lo que al ejecutarse varias veces obtendremos varias soluciones cuya diversidad se va controlando con el parámetro β.

Del algoritmo anterior han sido explicadas todos los pasos del método constructivo miope aleatorizado (I.1), pasos *1,2,3 a), b), c)* y *d*), quedando solamente pendiente la parte de la búsqueda local (denotado con *e*) en el algoritmo anterior) que describimos enseguida.

4.1.2 Búsqueda local

Las soluciones obtenidas con el método constructivo, pasos *3 a), b), c)* y *d*) del algoritmo anterior, se mejoran con un procedimiento de búsqueda local. Este es un procedimiento iterativo que en cada iteración se mueve de la solución actual a una

mejor solución de su vecindario. El procedimiento finaliza cuando no se encuentren mejores soluciones vecinas.

En este caso las soluciones vecinas se obtienen por cambios de cadenas de nodos de una ruta a otra (cambios entre-rutas) o en la misma ruta (cambios intra-rutas). En algunos casos se generalizan algunos movimientos que se utilizan en la heurística de inserción generalizada (GENI). Se consideran 3 tipos de cambios, uno intra-rutas y dos entre-rutas:

A) Cambio de una cadena dentro de una ruta.

B) Cambio de una cadena de una ruta a otra.

C) Intercambios de dos cadenas entre dos rutas diferentes.

Estos tres tipos de cambios están basados en dos operaciones, usadas de forma diferente en cada caso:

1) Eliminación de una cadena de una ruta.

2) Inserción de una cadena en una ruta.

En las siguientes tres figuras, sin pérdida de generalidad y para simplificar la notación, se identifican los puntos de visita con el orden que ocupan en la ruta. Los arcos que se eliminan se muestran con líneas de puntos, los que se añaden con guiones, y en los tramos que cambian de sentido se añade la nueva orientación debajo o a un lado de la original.

Específicamente se tienen 3 tipos de eliminaciones:

i) Eliminación usual ó clásica, mostrada en la figura 4.2. La eliminación de la cadena de nodos que se inicia en el punto i y contiene r puntos, consiste en la eliminación de los arcos $(i-1, i)$ y $(i+r-1, i+r)$, y la incorporación del arco $(i-1, i+r)$.

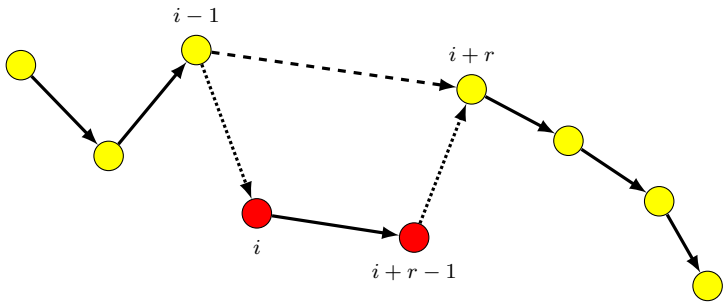


Figura 4.2: Eliminación clásica

ii) Eliminación tipo GENI-I, mostrada en la figura 4.3. La eliminación de la cadena de puntos que se inicia en el punto i y contiene r puntos consiste en la eliminación de los arcos $(i-1,i)$, $(i+r-1,i+r)$, $(k,k+1)$ y $(j,j+1)$; la incorporación de los arcos $(i-1,k)$, $(i+r,j)$, $(k+1,j+1)$; y el cambio de sentido de los tramos de $i+r$ a k, y de $k+1$ a j. Por lo tanto, además de los parámetros i y r que definen la cadena a eliminar, este tipo de eliminaciones depende de otros 2 parámetros j y k, $(k \geq i+r, j \geq k+1)$.

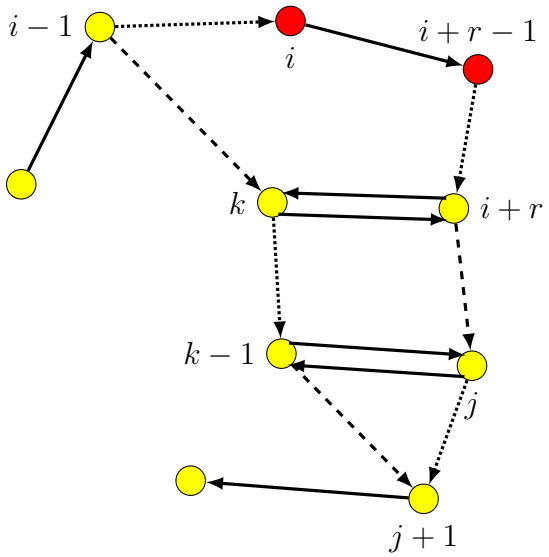


Figura 4.3: Eliminación tipo GENI-I

iii) Eliminación tipo GENI-II, mostrada en la figura 4.4. La eliminación de la cadena de puntos que se inicia en el punto i y contiene r puntos consiste en la

eliminación de los arcos $(i-1,i)$, $(i+r-1,i+r)$, $(j-1,j)$, $(l,l+1)$ y $(k,k+1)$; la incorporación de los arcos $(i-1,k)$, $(l+1,j-1)$, $(i+r,j)$, $(l,k+1)$; y el cambio de sentido de los tramos de $l+1$ a k, y de $i+r$ a $j-1$. Por tanto, además de i y r, este tipo de eliminaciones depende de 3 parámetros j, l y k, $(j \geq i+r+1, l \geq j, k \geq l+1)$.

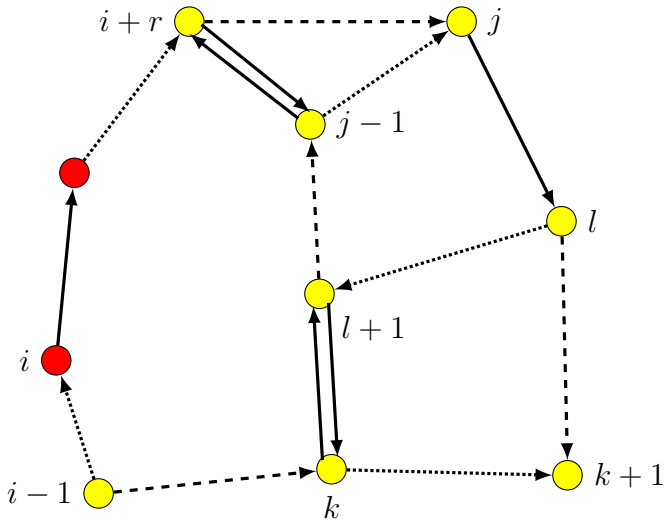


Figura 4.4: Eliminación tipo GENI-II

De manera similar, se consideran tres tipos de inserciones. En las tres siguientes figuras, igual que antes, se identifican los puntos de la ruta con el orden que ocupan. El primer y último punto de la cadena se identifican con x e y respectivamente. Los arcos que se eliminan se muestran con líneas de puntos, los que se añaden con guiones, y en los tramos que cambian de sentido se añade la nueva orientación debajo de la original.

i) Inserción usual ó clásica, mostrada en la figura 4.5. La inserción entre los puntos i e $i+1$, $(i \geq 1)$ de la cadena de vértices entre el x y y, consiste en la eliminación del arco $(i,i+1)$, y la incorporación de los arcos (i,x) e $(y,i+1)$.

ii) Inserción tipo GENI-I, que se muestra en la figura 4.6. La inserción en la ruta de la cadena de x a y consiste en la eliminación de los arcos $(i,i+1)$, $(j,j+1)$

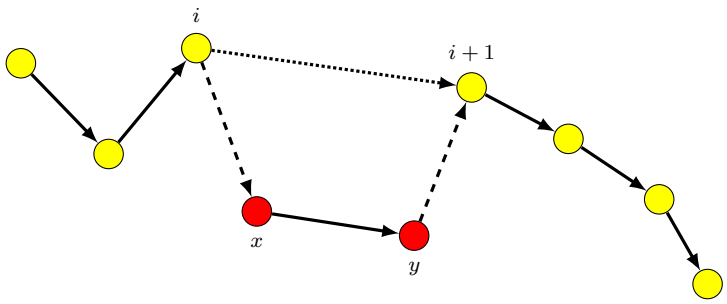


Figura 4.5: Inserción clásica

y $(k, k+1)$; la incorporación de los arcos (i, x), (y, j), $(i+1, k)$ y $(j+1, k+1)$; y el cambio de sentido de los tramos de $i+1$ a j, y de $j+1$ a k. Por tanto, además de la cadena a insertar, este tipo de inserciones depende de 3 parámetros i, j y k, $(i \geq 1, j \geq i+1, k \geq j)$.

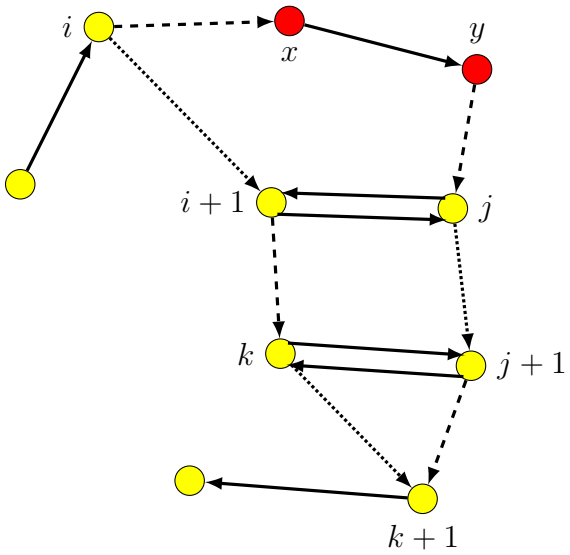


Figura 4.6: Inserción tipo GENI-I

iii) Inserción tipo GENI-II

La figura 4.7 ilustra dicha inserción que consiste en la eliminación de los arcos $(i, i+1)$, $(l-1, l)$, $(j, j+1)$ y $(k-1, k)$; la incorporación de los arcos (i, x), (y, j), $(l, j+1)$, $(k-1, l-1)$ y $(i+1, k)$; y el cambio de sentido de los tramos de l a j y de

$i+1$ a $l-1$. Por tanto, (además de la cadena a insertar) este tipo de inserciones dependen de 4 parámetros i, l, j y k, $(i \geq 1, l \geq i+2, j \geq l, k \geq j+2)$.

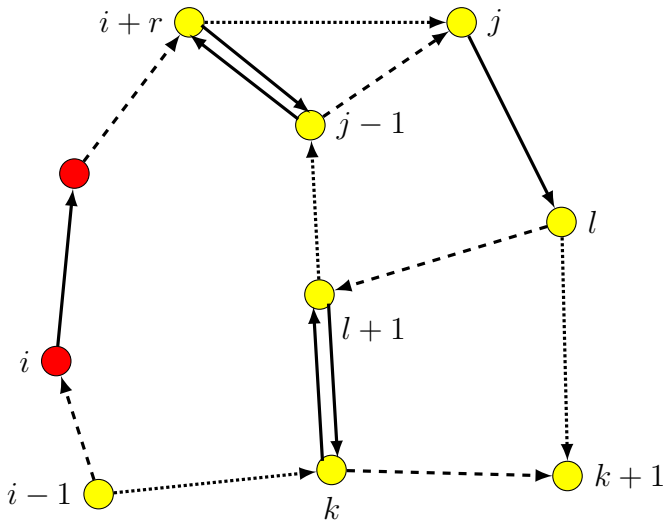


Figura 4.7: Inserción tipo GENI-II

Hay que indicar que tanto para las eliminaciones como para las inserciones se consideran ambas orientaciones de las rutas implicadas.

A continuación se explican los tres tipos de cambios que dan lugar a las diferentes soluciones vecinas.

A) Cambio de una cadena dentro de una ruta.

Este caso involucra la eliminación de una cadena de una ruta y su reinserción en la misma ruta. Para ello, la cadena seleccionada se elimina mendiante la eliminación clásica y a continuación se evalúan los tres tipos de inserciones: usual, GENI-I y GENI-II, en la ruta modificada.

B) Cambio de una cadena de una ruta a otra diferente.

Consiste en la eliminación de una cadena de una ruta y su inserción en otra distinta. Dada una cadena se evalúan los tres tipos de eliminaciones (usual,

GENI-I y GENI-II) y se considera la mejor de todas. A continuación se evalúan los tres tipos de inserciones en las otras rutas.

C) Intercambios de dos cadenas entre dos rutas diferentes.

En este caso se tiene la eliminación de las cadenas C_A y C_B de las rutas R_A y R_B respectivamente y la inserción de C_A en R_B y C_B en R_A. Sólo se considera la eliminación usual, pero sí se consideran los tres tipos de inserciones posibles.

Dentro de la evaluación de cada posible movimiento verificamos la factiblidad para satisfacer las restricciones del problema, esto es, no sobrepasar la capacidad de los vehículos y que la entrega se realice el día inicialmente solicitado o el día anterior (para $g = 1$).

Finalmente, es conveniente hacer notar lo siguiente.

- Tanto en las operaciones de eliminación como de inserción, se consideran las dos orientaciones de las rutas implicadas.
- Las eliminaciones tipo GENI-I incluyen a algunas de las eliminaciones usuales, concretamente cuando $k = i + r$ y $j = k + 1$. Sin embargo no abarca a todas, cuando la cadena a eliminar finaliza en el último o penúltimo punto antes de la vuelta al origen, la eliminación usual no puede ser incluida dentro del tipo GENI-I ya que ésta al menos necesita 2 puntos distintos (j y k) entre el final de la cadena y la vuelta al origen.
- La misma reflexión anterior se hace con las inserciones.
- En el caso de los cambios *A*) y *C*) las rutas de donde se eliminan cadenas van a ser modificadas posteriormente con alguna inserción. Por lo que no tiene mucho sentido buscar y ejecutar la mejor eliminación de una cadena de una ruta si después esa ruta va a ser modificada. Además, fijando la cadena (cambio *A*) o par de cadenas a cambiar (tipo *C*) y considerando cada ruta concreta, hay una dependencia de la inserción en esa ruta de la eliminación que ha habido antes en

la misma: la mejor combinación (inserción-eliminación) puede no corresponder a la mejor eliminación inicial. Sin embargo, el tiempo de cálculo para hallar esta mejor combinación puede ser excesivo dado la cantidad de parámetros que se manejan. Por lo tanto una solución razonable es considerar la eliminación usual.

- No ocurre lo mismo con el cambio tipo B), una vez que una cadena se elimina de su ruta, ésta no se modifica posteriormente. Por lo que para cada cadena, los procesos de eliminación e inserción son independientes. Entendemos que en este caso sí tiene sentido, y no supone un tiempo excesivo, elegir la mejor eliminación.

- Los cambios que se proponen generalizan las ideas expuestas en el trabajo de Gendreau et al. [41], donde se propone una heurística (GENIUS) para el TSP simétrico con buenos resultados.

Experimentos preliminares mostraron que la fase de búsqueda local descrita consume mucho tiempo de cómputo, por lo que se propone agilizar esta fase utilizando una estrategia denominada búsqueda local rápida [14] que se detalla enseguida.

Búsqueda local rápida

En el GRASP que hemos propuesto, una de las partes que consume más tiempo de computación es la correspondiente a la búsqueda local, ya que se exploran todos los posibles reacomodos de cadenas de vértices en la misma ruta o en las rutas diferentes. Se puede ahorrar mucho tiempo de computación si se sigue una idea propuesta por Bentley para el TSP [14], llamada búsqueda local rápida, la cual describimos a continuación.

La idea consiste básicamente en dividir el vecindario de la solución actual en subvecindarios más pequeños y asociar a cada uno de estos subvecindarios una variable binaria que indique si están activos o inactivos, de forma que en cada paso solamente se exploran los subvecindarios activos. Inicialmente todos los subvecindarios

están activos. En cada iteración, si al explorar un subvecindario no contiene alguna solución que mejore la actual pasa a ser inactivo, en caso contrario permanece activo. Por otra parte, si como resultado de un movimiento un subvecindario se modifica, entonces pasa a ser activo.

De esta forma a medida que el proceso avanza y la solución mejora, habrá menor número de subvecindarios activos y por tanto se utilizará menos tiempo de computación en exploraciones innecesarias sin que, obviamente, quede afectada la solución final. Esta pequeña modificación da lugar a lo que se denomina Búsqueda Local Rápida. En este sentido, cuanto mayor sea el número de subvecindarios que se consideren más tiempo de computación se consigue ahorrar como se ilustra en la figura 4.8. En ella se representa con el círculo grande de la izquierda un subvecindario grande de la solución actual sin subdividir, y con el círculo grande de la derecha el mismo subvecindario dividido en subvecindarios más pequeños. Los círculos pequeños en blanco representan soluciones que no mejoran la actual y el círculo pequeño en negro una que sí mejora la actual. Esta solución obliga, en el primer caso, a explorar todo el subvecindario grande en los siguientes pasos ya que estaría activo; mientras que en el segundo caso sólo es necesario explorar el subvecindario pequeño que contiene la solución que mejora la actual ya que los demás estarían inactivos.

En nuestro caso los subvecindarios que se consideran son de tres tipos, los del primer tipo contienen cada ruta de forma independiente, esto es, registran cambios al modificar alguna ruta, los denotamos por

$$A(1), A(2), \cdots, A(K)$$

ó

$$A(j), \quad j = 1, \ldots K$$

Los del segundo tipo registran los cambios que se realizan al mover cadenas de vértices de una ruta a otra,

$$B(j, j'), \quad j, j' = 1, \ldots, K, \quad j \neq j'$$

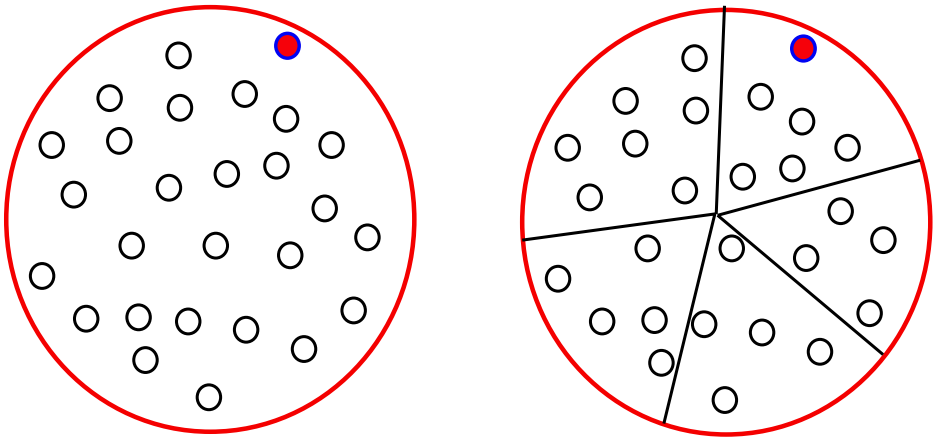


Figura 4.8: a)Subvecindario sin dividir: hay que explorarlo todo. b) Subvecindario dividido: no hay que explorar las partes que no contengan mejora

Los vecindarios del tercer tipo registran los intercambios de cadenas de vértices entre dos rutas diferentes

$$C(j, j'), \quad j = 1, \ldots K-1, \quad j' = j+1, \ldots, K, \quad j \neq j'$$

Otra modificación que realizamos fue tratar de agilizar todavía más la búsqueda local rápida, en lo que denominaremos Búsqueda local super rápida. La modificación consiste en que en cada subvecindario definido en la búsqueda local rápida se guarda no sólo si está activo o no, también se almacena el costo y los elementos del mejor movimiento que se puede hacer en este subvecindario, de tal forma que se pueda realizar la búsqueda local más rápidamente, con un costo de memoria superior pero ganando en tiempo.

En esto consiste la búsqueda local rápida y super rápida, como se mostrará en el capítulo 7, donde se exponen los resultados de los experimentos computacionales, se logra una gran reducción en los tiempos de cómputo sin afectar la calidad de las soluciones obtenidas.

Tenemos completa la descripción del GRASP incluyendo la búsqueda local, con lo que podemos generar soluciones diversas. Veremos enseguida cómo conectar

dos de estas soluciones buscando obtener soluciones aún mejores. Para ello debemos disponer de un mecanismo generador de trayectorias.

4.1.3 Encadenamiento de trayectorias

Con las ideas expuestas en las secciones anteriores, tenemos todos los procedimientos que nos permiten generar una variedad de soluciones mediante el GRASP diseñado.

Con las mejores soluciones obtenidas en la fase I, formamos el conjunto $SetofBest$. La segunda fase, el encadenamiento de trayectorias, consiste en combinar cada par de estas buenas soluciones buscando obtener soluciones de mejor calidad.

Sean S_1 y $S_2 \in SetofBest$, construiremos una trayectoria con soluciones entre S_1 y S_2, esto es, partiendo de S_1 realizamos movimientos para llegar a S_2 y exploramos el espacio de búsqueda en la trayectoria entre estas dos soluciones elite.

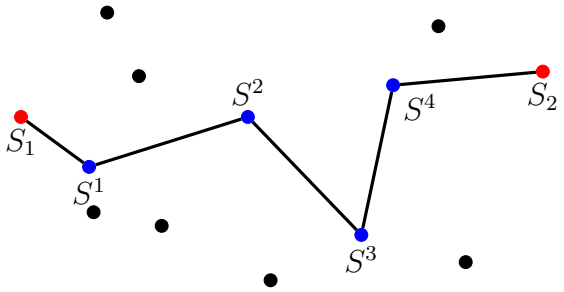


Figura 4.9: Trayectoria entre las soluciones S_1 y S_2

Para la generación de trayectorias debemos partir de una solución, S_1, y generar una sucesión de soluciones intermedias $S^1, S^2, S^3, \ldots$, de tal forma que en cada nueva solución intermedia se incorporen atributos de S_2 que no estén presentes en S_1.

Veamos los detalles para generar las trayectorias. Para la generación de las soluciones intermedias utilizaremos algunas ideas de Beasley [12] usadas recientemente por Prins [80]. Consideraremos cada solución como una gran cadena, esto es, a cada solución S le asignamos una gran cadena BC. Inicialmente se construyen dos

grandes cadenas con todos los nodos correspondientes a S_1 y S_2, llamémosles, BC_1 y BC_2, respectivamente.

Iniciando en BC_1 se realizan pequeños cambios, de forma iterativa, hasta obtener BC_2. De esta forma obtendremos una sucesión de grandes cadenas $BC^1, BC^2, \ldots$, en cada iteración cada gran cadena obtenida es más similar a BC_2 que la gran cadena anterior. Para cada una de estas grandes cadenas intermedias se obtiene una solución dividiendo la gran cadena en subcadenas correspondientes a rutas factibles, obteniendo las soluciones intermedias $S^1, S^2, S^3, \ldots$. A partir de estas soluciones aplicamos un proceso de búsqueda local para mejorarlas. El bosquejo del algoritmo es el siguiente.

Algoritmo Fase II: Encadenamiento de trayectorias

1. Generar BC_1 y BC_2 las grandes cadenas asociadas a S_1 y S_2.
2. Generar la sucesión de cadenas intermedias $BC_1 = BC^0, BC^1, BC^2, \ldots, BC^q = BC_2$
3. Asociar a cada gran cadena intermedia, BC^i, una solución factible S^i
4. Aplicar una búsqueda local partiendo de cada S^i

Note que los pasos 1, 2 y 3 corresponden a II.1 en el diagrama de la fase II de la página 70, mientras que 4 corresponde a II.2.

El proceso de búsqueda local que se utiliza es el proceso descrito en la sección 4.1.2. Veremos enseguida los detalles para la formación de las grandes cadenas, la generación de la sucesión de cadenas intermedias y la obtención de soluciones factibles a partir de las grandes cadenas intermedias generadas.

Formación de una gran cadena a partir de una solución factible

A cada solución factible, S, le asignamos una gran cadena BC, esto se hace ensamblando las rutas sin considerar el depósito. Por ejemplo, si las soluciones S_1 y S_2 están dadas por las rutas

$$S_1: \quad \{0,1,4,2,3,0\},\{0,5,6,8,7,0\}$$

y

$$S_2: \quad \{0,1,2,6,8,0\},\{0,5,4,3,7,0\}$$

entonces sus correspondientes grandes cadenas están dadas por

$$BC_1 = 1-4-2-3-5-6-8-7$$

y

$$BC_2 = 1-2-6-8-5-4-3-7$$

La relación entre grandes cadenas a soluciones está bien definida, ya que dada una solución factible, está claro que se puede formar una única cadena.

Generación de la sucesión de grandes cadenas intermedias

Los movimientos para generar la sucesión de soluciones intermedias se realizan en las grandes cadenas asociadas a cada solución, esto es, se genera una sucesión de grandes cadenas intermedias.

Partiendo de BC_1 para la obtención de las grandes cadenas intermedias, en cada paso se realizan cambios con el fin de aumentar el número de arcos que tiene en común con BC_2; específicamente se verifican los intercambios entre dos cadenas consecutivas de vértices hasta encontrar el primer intercambio que incremente el número de arcos en común con BC_2.

El siguiente ejemplo ilustra esta idea. Consideremos las grandes cadenas anteriores,

$$BC_1 = 1-4-2-3-5-6-8-7, \qquad BC_2 = 1-2-6-8-5-4-3-7$$

con el proceso descrito se obtiene la siguiente sucesión de grandes cadenas

$$\begin{aligned} BC^0 = BC_1 &= 1-4-2-3-5-6-8-7 \\ BC^1 &= 1-4-2-6-8-3-5-7 \\ BC^2 &= 1-2-6-8-3-5-4-7 \\ BC^3 = BC_2 &= 1-2-6-8-5-4-3-7 \end{aligned}$$

Como puede observarse BC^0 tiene en común con BC_2 el arco (6,8); BC^1 tiene en común con BC_2 los arcos (2,6) y (6,8) y BC^2 tiene en común con BC_2 los arcos (1,2), (2,6), (6,8) y (5,4). Finalmente, $BC^3 = BC_2$.

También, BC^1 se obtiene de BC^0 intercambiando 3-5 y 6-8; BC^2 se obtiene de BC^1 intercambiando 4 y 2-6-8-3-5; y BC^3 se obtiene de BC^2 intercambiando 3 y 5-4.

Cada gran cadena tendrá asociada una solución factible, lo que da lugar a la generación de la sucesión de soluciones intermedias entre S_1 y S_2, dentro del espacio solución. Sólo nos falta describir este paso, esto es, la obtención de una solución factible a partir de una gran cadena.

Obtención de una solución factible a partir de una gran cadena

El último paso por detallar es cómo se obtiene una solución factible a partir de una gran cadena. Para ello se requiere dividir la gran cadena en subcadenas que correspondan a rutas factibles, por ejemplo, supongamos que

$$BC^1 = 1-4-2-6-8-3-5-7$$

entonces se puede obtener una solución S^1 dada por

$$S^1: \quad \{0,1,4,2,0\}, \{0,6,8,3,5,7,0\}$$

si se divide la gran cadena en el arco 2-6, siempre que las rutas $\{0,1,4,2,0\}$ y $\{0,6,8,3,5,7,0\}$ satisfagan las restricciones del problema.

Para lograr la subdivisión de la gran cadena en rutas factibles de forma óptima se procede de la siguiente forma [12]. Consideremos la gran cadena

$$b_1 - b_2 - \cdots - b_n$$

Por simplicidad definamos $b_0 = 0$. Consideremos el grafo auxiliar $GR = (Nodo, Arco)$ donde $Nodo = \{0, 1, 2, \ldots, n\}$ es el conjunto de nodos y $Arco = \{(i, j) : i < j, i, j \in Nodo\}$. Definamos también para todo $(i, j) \in Arco$

$$u_{ij} = \begin{cases} \text{Costo de la ruta que inicia en 0, visita los nodos } b_{i+1}, b_{i+2}, \ldots, b_j, \\ \quad \text{y regresa a 0,} \quad \text{si la ruta es factible} \\ \infty \text{ en otro caso} \end{cases}$$

Para obtener una división óptima en esta gran cadena, esto es, el conjunto de subcadenas correspondientes a rutas con mínimo costo, podemos considerar el problema de la ruta más corta del nodo 0 al nodo n en el grafo, GR, con los costos definidos en u_{ij}. Por ejemplo, si la solución del problema de la ruta mas corta es $0 - 4 - 7 - n$, entonces las rutas obtenidas son las siguientes

$$\{0, b_1, b_2, b_3, b_4, 0\}, \{0, b_5, b_6, b_7, 0\}, \{0, b_8, b_9, \ldots b_{n-1}, b_n, 0\}$$

Existen algoritmos rápidos y eficientes para resolver el problema de la ruta más corta, uno de los más conocidos y que ha probado su efectividad es el algoritmo de Dijkstra [2].

De esta forma se obtienen las soluciones factibles asociadas a cada una de las grandes cadenas intermedias.

Con estos elementos se generan las trayectorias entre soluciones elite.

Con esto queda completa la descripción de la primera metaheurística propuesta para el problema mono-objetivo, el algoritmo HGET, híbrido de GRASP y encadenamiento de trayectorias. Describimos enseguida la segunda propuesta basada en una búsqueda por entornos variables, que utiliza algunos de los elementos aquí descritos.

4.2 Otra metodología de solución: Búsquedas por entornos variables (BEV-RF)

Presentamos enseguida otra propuesta de metodología de solución para el problema mono-objetivo. Esta se desarrolla ya que uno de los propósitos de este trabajo es abordar el problema desde la perspectiva multi-objetivo minimizando costos de transportación y de almacenamiento, y esta metaheurística nos servirá de base para la metodología que hemos diseñado para abordar el problema biobjetivo.

Esta segunda metodología de solución que proponemos está basada en una búsqueda por entornos variables, metaheurística desarrollada por Hansen y Mladenović [49] y descrita en 2.2.1. Como se explicó en el capítulo 2 esta metaheurística realiza búsquedas locales utilizando un cambio sistemático de vecindades o entornos.

En la descripción utilizaremos algunos de los elementos introducidos en las secciones anteriores para definir las estructuras de los vecindarios. Se propone utilizar un procedimiento de búsqueda por entornos variables básica, que combina una componente aleatoria y una fase de búsqueda local.

Si denotamos por $f(S)$ el valor de la función objetivo en una solución S, el algoritmo para resolver nuestro problema puede ser descrito en la siguiente forma.

Algoritmo BEV-RF

1. Construir una solución inicial $S = S_0$, $niter = 0$
2. Repetir

   $niter = niter + 1$

   $r = 0$

   Repetir

   $r = r + 1$

   Obtener $S_1 =$ **Agitación**(r, S)

Aplicar **Búsqueda local** a partir de S_1

Hasta que $r = r_{\text{máx}}$ o $f(S_1) < f(S)$

Si $f(S_1) < f(S)$ entonces $S = S_1$

Hasta satisfacer el criterio de parada

La solución inicial se obtiene por el método constructivo miope aleatorizado del algoritmo GRASP descrito en la sección 4.1.1. La búsqueda local que se utiliza es la descrita en la sección 4.1.2. Veremos enseguida a detalle la descripción del procedimiento de agitación.

4.2.1 Procedimiento de agitación

En este procedimiento se genera una solución que pertenece a una vecindad de la solución actual, S. Depende de un parámetro, r, de forma tal que valores pequeños de r tendrán asociados vencindarios pequeños, es decir, se generan soluciones cercanas o parecidas a S; mientras que para valores altos de r se tendrán asociados vecindarios más grandes, es decir, soluciones más alejadas a S.

La agitación en nuestra implementación está basada en la idea de la asociación de soluciones a grandes cadenas, lo cual fue descrito a detalle en la sección 4.1.3. Se trabaja sobre las grandes cadenas, dividiéndolas aleatoriamente en subcadenas, se permutan estas subcadenas también de forma aleatoria, se reensamblan en una nueva gran cadena y a partir de ésta se construye una solución factible, que será el resultado del procedimiento de agitación.

El algoritmo puede ser descrito como sigue.

Procedimiento de agitación

Entrada(S, r)**,** **Salida**(S_1)

1. Obtener una gran cadena BC ensamblando todas las rutas en S sin considerar el depósito
2. Dividir la gran cadena en $(r+1)$ subcadenas generando aleatoriamente r puntos de corte en BC
3. Permutar aleatoriamente las subcadenas y reensamblarlas, para formar una nueva gran cadena BC'
4. Obtener una solución a partir de BC' subdividiendola en un conjunto de rutas factibles.

Para ejemplificar el procedimiento de agitación, sea S una solución factible y supongamos $r = 2$.

$$S: \quad \{0,1,4,2,3,0\}, \{0,5,6,8,7,0\}$$

Entonces formamos su gran cadena

$$BC = 1-4-2-3-5-6-8-7$$

y como $r = 2$, se formarán 3 subcadenas, sean por ejemplo

$$1-4-2, \qquad 3-5, \qquad 6-8-7$$

Estas subcadenas se permutan aleatoriamente, supongamos que resultan

$$3-5, \qquad 6-8-7, \qquad 1-4-2,$$

Se reensamblan para formar una nueva gran cadena,

$$BC' = 3-5-6-8-7-1-4-2$$

Esta nueva gran cadena se convierte en una solución factible con el proceso descrito en la sección 4.1.3 mediante la construcción de un grafo auxiliar y la definición sobre él del correspondiente problema de la ruta más corta.

Esto completa la descripción del procedimiento de agitación y del algoritmo de búsqueda por entornos variables.

Capítulo 5

El problema biobjetivo

Recordemos que el problema que estamos trabajando consiste en diseñar las rutas en todo el horizonte de planeación, buscando minimizar la distancia total recorrida y satisfaciendo la demanda de los distribuidores. La propuesta desarrollada en los capítulos anteriores para resolver el problema planteado es minimizar los costos de transportación permitiendo que las entregas de los productos puedan realizarse en días anteriores a los inicialmente propuestos por las delegaciones. Con esto disminuirán los costos de transportación, pero a la vez se ocasionan costos de stock o de almacenamiento, ya que al adelantar pedidos éstos tienen que almacenarse hasta el día que se utilizarán.

Si bien en el caso real que motivó este estudio, el costo ocasionado por el almacenamiento de los productos durante un día era perfectamente aceptable para la compañía, en situaciones más generales, donde se permita un mayor adelanto, debe estudiarse el impacto de este costo de almacenamiento. Por consiguiente en este capítulo estudiaremos el problema desde la perspectiva multi-objetivo; veremos cómo encontrar un equilibrio entre los costos de transportación y los de almacenamiento, buscando buenas soluciones compromiso para ambos objetivos.

En la primera sección proponemos un modelo matemático del problema que consiga minimizar los dos objetivos simultáneamente, el cual se basa en el modelo II presentado en el capítulo 3 para el problema mono-objetivo. En la segunda sección se describen los principales elementos de la optimización multi-objetivo que serán necesarios para desarrollar el método de solución que proponemos. En la sección tres

se valida el modelo propuesto resolviendo algunas instancias pequeñas con métodos tradicionales de optimización multi-objetivo.

5.1 Un modelo biobjetivo

En los capítulos anteriores mostramos cómo resolver el problema de minimizar los costos de transportación, lo cual se hace considerando el problema de ruteo sobre todo el horizonte de planeación y permitiendo cierta flexibilidad en la fecha de entrega de productos. Con esto se logra efectivamente disminuir estos costos, pero para la compañía esto repercute en que en algunas delegaciones tendrán que almacenar los productos uno (o varios días) antes de su utilización.

Entonces se plantea el problema de tratar de encontrar un equilibrio entre estos dos costos, esto es, minimizar los costos de transportación sin que esto implique un alto costo de almacenamiento. Presentamos enseguida una propuesta de modelación que involucra ambos objetivos.

Tomaremos como base el modelo mono-bjetivo II planteado en el capítulo 3, el cual minimiza sólo los costos de transportación. Partimos de este modelo base ya que es el que controla el número de vehículos por día o por cada período en el horizonte de planeación. Recordemos la notación utilizada en el capítulo 3.

$$
\begin{aligned}
HT &= \text{Número de días en el horizonte de planeación} \\
n &= \text{número total de órdenes} \\
V &= \{0, 1, 2, \ldots, n\} \text{ conjunto de órdenes, 0 corresponde al depósito} \\
K &= \text{número máximo de vehículos disponibles en un período del horizonte de planeación} \\
Q &= \text{capacidad de los vehíıculos} \\
c_{ij} &= \text{distancia entre la locación de la orden } i \text{ y la locación de la orden } j, \ \forall i, j \in V \\
q_i &= \text{demanda de la orden } i, \ \forall i = 1, \ldots, n
\end{aligned}
$$

e_i = deadline de la orden i, $\forall i = 1, \ldots, n$

g = número máximo de días que una orden puede ser servida antes de su *deadline*

Las variables de decisión consideradas son las siguientes:

$$x_{ij} = \begin{cases} 1, & \text{si la locación } j \text{ se visita justo después de la locación } i \\ 0, & \text{en otro caso} \end{cases}$$

$\forall i, j \in V$.

$$y_{id} = \begin{cases} 1, & \text{si la locación } i \text{ se visita el día } d \\ 0, & \text{en otro caso} \end{cases}$$

$\forall i \in V\backslash\{0\}, \forall d = 1, \ldots, HT$.

$$z_{id} = \begin{cases} 1, & \text{si la locación } i \text{ se visita el día } d \text{ y es la primera de su ruta} \\ 0, & \text{en otro caso} \end{cases}$$

$\forall i \in V\backslash\{0\}, \forall d = 1, \ldots, HT$.

El modelo II propuesto en el capítulo 3 con un solo objetivo es el siguiente.

$$\text{mín} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

sujeto a

$$\sum_{i \in V} x_{ij} = 1, \qquad \forall j \in V\backslash\{0\} \tag{5.1}$$

$$\sum_{j \in V} x_{ij} = 1, \qquad \forall i \in V\backslash\{0\} \tag{5.2}$$

$$\sum_{i \in V} x_{i0} = \sum_{j \in V} x_{0j} \tag{5.3}$$

$$u_i - u_j + Q x_{ij} \leq Q - q_j, \qquad \forall i, j \in V\backslash\{0\}, i \neq j, \text{ tal que } q_i + q_j \leq Q \tag{5.4}$$

$$q_i \leq u_i \leq Q, \qquad \forall i \in V\backslash\{0\} \tag{5.5}$$

$$y_{id} \leq (mt)_{id}, \qquad \forall i \in V\backslash\{0\}, \forall d \in \{1, 2, \ldots, HT\} \tag{5.6}$$

$$\sum_{d=1}^{HT} y_{id} = 1, \qquad \forall i \in V\backslash\{0\} \tag{5.7}$$

$$y_{id} - y_{jd} \leq 1 - x_{ij}, \qquad \forall i, j \in V\backslash\{0\}, i \neq j \tag{5.8}$$

$$z_{id} \geq y_{id} + x_{0i} - 1, \qquad \forall i \in V\backslash\{0\}, \forall d \in \{1, 2, \ldots, HT\} \tag{5.9}$$

$$\sum_{i=1}^{n} z_{id} \leq K, \qquad \forall d \in \{1, 2, \ldots, HT\} \tag{5.10}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in V \tag{5.11}$$

$$y_{id} \in \{0, 1\}, \qquad \forall i \in V\backslash\{0\}, \forall d \in \{1, 2, \ldots, HT\} \tag{5.12}$$

$$z_{id} \geq 0, \qquad \forall i \in V\backslash\{0\}, \forall d \in \{1, 2, \ldots, HT\} \tag{5.13}$$

donde mt_{id} es un parámetro calculado que toma el valor de 1 si la orden i puede ser servida en el día d, y 0 en otro caso.

Por comodidad se cambió la numeración de acuerdo a este capítulo y así se hará referencia a estas restricciones, pero corresponden al mismo modelo planteado en el capítulo 3.

Denotaremos como $\mathcal{X}$ el conjunto de soluciones factibles, esto es, el conjunto de soluciones que satisfacen las restricciones (5.1) a (5.13). Tomando como base este modelo, trabajaremos ahora sobre el modelo de dos objetivos, esto es, tratar de minimizar los costos de transportación y el almacenamiento simultáneamente.

Sea S una solución factible, denotamos por $f_1(S)$ la distancia total recorrida por las rutas en esa solución, esto es,

$$f_1(S) = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

este es el objetivo del costo de transportación el cual está ya bien definido desde el capítulo 3; nos enfocaremos a modelar el costo de stock o almacenamiento.

Tenemos que $y_{id} = 1$ nos indica que d es el día de entrega de la orden i. Si $d \neq e_i$ entonces la orden i se entrega antes de la fecha requerida y es necesario el almacenamiento de q_i productos durante $(e_i - d)$ días . Por ejemplo, supongamos que la orden i tiene $e_i = 4$, se permite un día de adelanto y $y_{i3} = 1$, esto es, la entrega se

realiza un día antes. Entonces tendremos que es necesario un día de almacenamiento para la orden i, por lo que se requiere almacenar q_i productos en la locación i.

Por consiguiente si $y_{id} = 1$ el almacenamiento en el nodo i, que denotaremos por h_i viene dado por la expresión

$$h_i = \sum_{d \in HT} (e_i - d) q_i y_{id}$$

Note que sólo una de las y_{id} será diferente de cero para cada orden i, ya que se tiene la restricción de que la entrega de la orden se realice solamente un día.

Para contabilizar el almacenamiento total, tenemos que sumar los productos que se almacenan en cada una de las órdenes, entonces el segundo objetivo, $f_2(S)$, está dado por

$$\begin{aligned} f_2(S) &= \sum_{i \in V} h_i \\ &= \sum_{i \in V} \sum_{d \in HT} (e_i - d) q_i y_{id} \end{aligned}$$

Teniendo en cuenta todo lo anterior, el modelo biobjetivo que proponemos es el siguiente:

$$\min_{x \in \mathcal{X}} F(x) \quad \text{donde} \quad F = (f_1, f_2)$$

$f_1(S) = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$ es el costo de la distancia recorrida por todos los vehículos

y

$f_2(S) = \sum_{i \in V} \sum_{d \in HT} (e_i - d) q_i y_{id}$ es el total de productos almacenados.

Con las restricciones (5.1) a (5.13).

A continuación exponemos algunos elementos básicos de optimización multiobjetivo que necesitaremos para resolver el modelo biobjetivo.

5.2 Algunos conceptos de optimización multi-objetivo

La mayoría de los problemas de decisión reales implican la optimización simultánea de más de un criterio. En estos casos, generalmente no existe una alternativa que sea la mejor opción para cada uno de los criterios al mismo tiempo y se requiere un análisis para determinar qué alternativa o alternativas son las más adecuadas para este problema. La programación multi-objetivo es la rama de la programación matemática que se ocupa de este tipo de problemas [31].

Un problema multi-objetivo (Multiobjective Problem: MOP) puede definirse como

$$MOP = \begin{cases} \min F(x) \\ sujeto \;\; a \;\; x \in \mathcal{X} \end{cases}$$

donde

$$F(x) = (f_1(x), f_2(x), \ldots, f_p(x))$$

$p \geq 2$ es el número de funciones objetivo, x son las variables de decisión, $\mathcal{X}$ es el espacio de soluciones factibles y $F(x)$ el vector de objetivos.

El conjunto $\mathcal{Y} = F(\mathcal{X})$ corresponde a las soluciones factibles en el espacio de objetivos ó espacio imagen y $y = (y_1, y_2, \ldots, y_p)$, $y_i = f_i(x)$ es una solución en el espacio de objetivos.

La primera dificultad con la que nos encontramos al afrontar un problema de programación multi-objetivo es cómo decidir cuándo una elección es más deseable que otra. Debido a que no tenemos una función objetivo, sino un vector de funciones objetivo, no podremos determinar siempre cuándo un vector es mayor o menor que otro, puesto que en $\Re^n$ no existe un orden total. Se introduce un orden inspirado en el concepto de eficiencia desarrollado por Vilfredo Pareto en 1896 de forma que la solución de un problema multi-objetivo ha de ser aquella, o estar entre aquellas, en las que no se pueda mejorar un objetivo sin empeorar alguno de los otros.

La dominancia en el sentido de Pareto, se define como sigue:

Una solución $y = (y_1, y_2, \ldots, y_p)$ domina (denotado por $\prec$) a una solución $z = (z_1, z_2, \ldots, z_n)$ si y sólo si $\forall i \in \{1, 2, , \ldots, p\}$, $y_i \leq z_i$ y existe $j \in \{1, 2, \ldots, p\}$ para el cual $y_j < z_j$. Nos interesarán aquellas soluciones x' cuyas imágenes $F(x')$ sean no dominadas.

El orden de Pareto es un orden parcial y no se puede elegir un punto como preferido sobre todos los demás. Entonces, es necesario desarrollar otro concepto, que generalice el concepto de optimalidad, y que permita señalar, en un problema multi-objetivo, los puntos que constituyan una buena elección: necesitamos el concepto de eficiencia. Enseguida veremos este y otros conceptos importantes [31].

Una solución factible $x' \in \mathcal{X}$ se llama eficiente u óptima de Pareto, si no existe otra $x \in \mathcal{X}$ tal que $f_i(x) \leq f_i(x')$, $\forall i \in \{1, 2, \ldots, p\}$ con al menos un $j \in \{1, 2, \ldots, p\}$ tal que $f_j(x) < f_j(x')$

El conjunto de todas las soluciones eficientes $x' \in \mathcal{X}$ se denota por $\mathcal{X}_E$

El conjunto de todos los puntos no dominados $y' = F(x') \in \mathcal{Y}$ donde $x^i \in \mathcal{X}_E$ se denota por $\mathcal{Y}_N$

Se definen en el espacio imagen el punto ideal y el punto de Nadir, que son tomados como puntos de referencia en algunas técnicas de solución de problemas multi-objetivo.

Punto Ideal: $y^I \in \Re^p$ donde

$$y_k^l = \min_{x \in \mathcal{X}} f_k(x) = \min_{y \in \mathcal{Y}} y_k$$

Este vector contiene los mejores valores individuales para cada objetivo.

Punto Nadir: $y^N \in \Re^p$ donde

$$y_k^N = \max_{x \in \mathcal{X}_E} f_k(x) = \max_{y \in \mathcal{Y}_N} y_k$$

Gran parte de las técnicas de programación multi-objetivo se centran en obtener el conjunto de puntos eficientes $\mathcal{X}_E$ del problema, o bien una aproximación a él. Se trata de encontrar un conjunto de puntos eficientes suficientemente amplio y representativo como para poder encontrar en él una alternativa que se ajuste a las preferencias del decisor.

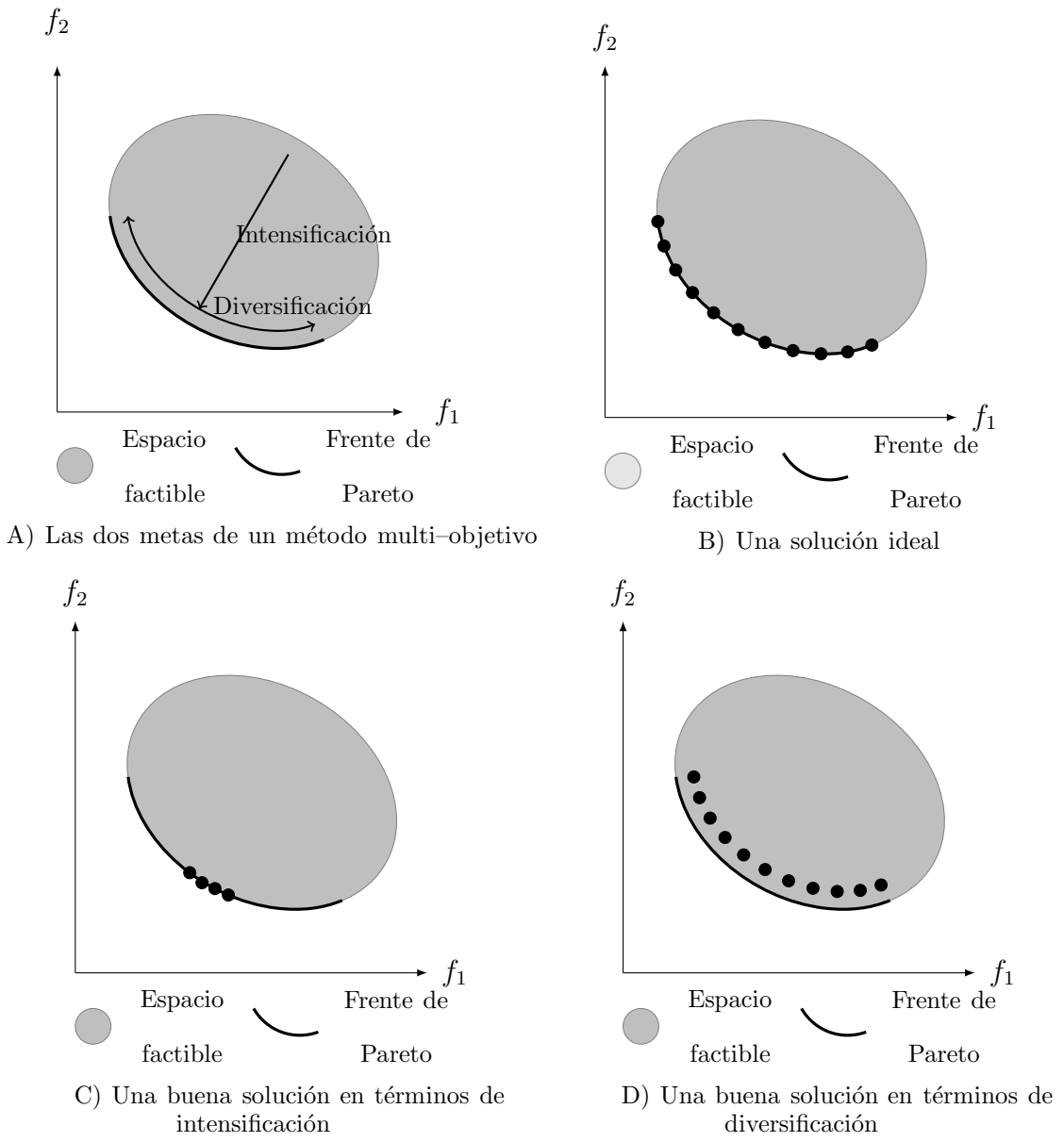
En problemas multi-objetivo, los métodos que aproximan el conjunto eficiente deben ser capaces de converger al conjunto de Pareto y al mismo tiempo generar un conjunto de soluciones diversas en el espacio de objetivos. En optimización mono-objetivo la diversidad se mide con referencia al espacio solución, esto es, la diversidad aumenta cuando las soluciones tienen diferentes propiedades estructurales, mientras que en optimización multi-objetivo se pretende encontrar soluciones que son diversas en el espacio de funciones objetivo, esto es, se pretende encontrar un frente de Pareto denso y distribuido en la región de interés.

Intensificación y diversificación es lo que se busca al aproximar la frontera de Pareto. Estos dos objetivos se ilustran en la figura 5.2A). La figura 5.2B) muestra una aproximación que cumple con estos dos objetivos, mientras que en la figura 5.2C) la aproximación es buena en términos de intensificación pero no es buena en diversificación, en la figura 5.2D) se ilustra el caso contrario.

Entonces retornando a nuestro problema, lo que se desea es encontrar un conjunto de soluciones eficientes, que sean de buena calidad en cuanto a costos de transportación y de almacenamiento.

5.3 Validación del modelo biobjetivo

Los dos objetivos planteados en nuestro problema se encuentran en conflicto, como lo ejemplificaremos más adelante. Si minimizamos sólo $f_1(S)$ entonces el modelo tratará de generar rutas que adelanten pedidos para bajar los costos y esto generará un almacenamiento alto. Por otro lado, si sólo minimizamos $f_2(S)$ entonces,



A) Las dos metas de un método multi–objetivo

B) Una solución ideal

C) Una buena solución en términos de intensificación

D) Una buena solución en términos de diversificación

Figura 5.1: Diversificación e Intensificación

al no tomar en cuenta los costos de transporte, se generarán rutas con una distancia recorrida muy alta, pero con un almacenamiento de cero entregando todos los pedidos el día originalmente solicitado por las delegaciones.

Tenemos que generar un conjunto de soluciones adecuadas a los dos objetivos. Generaremos el Frente de Pareto, esto es, las soluciones eficientes o no dominadas, soluciones que no mejoren uno de los objetivos sin empeorar el otro.

En nuestro contexto, si S y S' son dos soluciones, se dice que S domina a S' si

1. $f_1(S) \leq f_1(S')$
2. $f_2(S) \leq f_2(S')$ y
3. $f_1(S) < f_1(S')$ o bien $f_2(S) < f_2(S')$

Para validar nuestro modelo resolveremos algunas instancias pequeñas con métodos tradicionales y software comercial para verificar que se obtienen soluciones eficientes con el modelo propuesto.

5.3.1 Métodos tradicionales aplicados al modelo

Validaremos el modelo propuesto resolviendo con software comercial algunas instancias de prueba para verificar que se satisfacen las condiciones establecidas. Primero hay que hacer notar que los dos objetivos no se miden bajo la misma métrica, esto es, las unidades de medida de cada uno de ellos son diferentes, al igual que las magnitudes que éstos manejan. Por ejemplo, la distancia total que recorren algunos vehículos puede estar dada en miles de kilómetros aún en instancias con pocas órdenes, en tanto que el almacenamiento estará acotado por $g \sum_{i \in V} q_i$ en el caso extremo que todos los pedidos se adelanten g días, esto es, que tengamos que almacenar todos los productos g días. $f_1(S)$, en general, se medirá en magnitudes mucho más grandes que $f_2(S)$. Por ello, para la aplicación de algunos métodos, sería necesario normalizar primeramente ambos objetivos.

Método de la ε–restricción

Resolveremos algunas de las instancias que se resolvieron con el modelo mono-objetivo y descritas en la sección 3.1.1. Utilizaremos el método de ε–restricción para resolver el problema con dos objetivos. Este es uno de los métodos más utilizados en programación multi-objetivo y su implementación computacional en nuestro caso es sencilla ya que el almacenamiento se puede medir de forma discreta sobre un conjunto finito de valores.

El método de la ε–restricción para el caso biobjetivo consiste, a grandes rasgos, en optimizar un solo objetivo considerando el otro objetivo como una restricción acotada por un cierto valor ε. Variando el valor de tendremos varias soluciones eficientes de donde obtendremos una aproximación de la frontera de Pareto.

Como los valores de $f_2(S)$ son discretos y su intervalo de variación está acotado, entonces una forma de aplicar el método de ε–restricción a nuestro problema es considerar un proceso iterativo, minimizando sólo la distancia total recorrida agregando la restricción de acotar el almacenamiento.

Para ello primeramente resolveremos

$$\min_{\mathcal{X}} f_1(S)$$

Esto nos dará el menor valor de la distancia. Para esta solución evaluamos $f_2(S)$, el almacenamiento y esto nos dará un valor entero, digamos γ , entonces podemos resolver el siguiente problema para diferentes valores de h.

$$\begin{gathered}\min_{\mathcal{X}} f_1(S)\\ \text{sujeto a}\\ f_2(S) \leq \gamma - h\end{gathered}$$

Si γ es pequeño podemos ir bajando el valor de h de 1 en 1 ó en pasos más grandes si γ es grande. La idea es ir obteniendo las soluciones minimizando sólo un objetivo. Como $f_2(S)$ toma sólo valores enteros, si bajamos de 1 en 1 obtendremos todas las soluciones de la frontera de Pareto.

Aplicaremos esta técnica a las instancias de 10 y 12 nodos descritas en el capítulo 3, con 2 vehículos disponibles por día y permitiendo un día de adelanto para la entrega. Se toman estas instancias pequeñas ya que como se ha mencionado, el problema de minimizar sólo la distancia es NP-completo y tendremos que resolverlo varias veces, por lo que optamos por instancias chicas para obtener resultados en poco tiempo.

Todos los experimentos se resolvieron utilizando Cplex v. 11.1 en un servidor Sun Fire V440 con 4 procesadores Ultra Sparc III a 1062 GHz con 8Gb de RAM. Cplex se ejecutó con los parámetros por default, en particular con una tolerancia de error relativo de 10^{-4} para el optimizador de problemas entero mixtos.

Para el ejemplo de 10 nodos, si minimizamos sólo la distancia obtenemos un valor óptimo de 413 y con esto se genera un almacenamiento de 28. Resolviendo ahora

$$\min_{\mathcal{X}} f_1(S)$$
$$sujeto$$
$$f_2(S) \leq 27$$

Obtenemos que el valor mínimo de la distancia con esta restricción adicional es ahora 431 y evaluando el almacenamiento, obtenemos que es necesario almacenar 23 unidades. Siguiendo este esquema obtenemos los siguientes resultados.

f_1	413	431	436	437	455	485	504	553
f_2	28	23	19	17	12	11	4	0

Si optimizamos sólo $f_2(S)$, esto es, sólo el almacenamiento sin considerar la distancia, obtenemos que el almacenamiento alcanza su valor óptimo en 0 con un valor de la distancia de 667.

El punto ideal y el punto de Nadir para este ejemplo están dados por

Punto Ideal = $(413, 0)$

Punto Nadir = $(553, 28)$

Aquí se observa claramente que ambos objetivos se encuentran en conflicto, minimizar uno de ellos aumenta el valor del otro y viceversa.

Para el caso de 12 nodos, se procede de forma similar, si optimizamos solo la distancia obtenemos un valor de 445 con 33 unidades de almacenamiento. Resolviendo

$$\min_{\mathcal{X}} f_1(S)$$
$$sujeto$$
$$f_2(S) \leq 32$$

Se obtiene un valor de la distancia de 459 y 23 unidades de almacenamiento. Los valores obtenidos se presentan enseguida.

f_1	445	459	464	483	522	532	541	590
f_2	33	23	19	12	11	8	4	0

Si sólo optimizamos el almacenamiento, obtenemos como era de esperarse que el óptimo se alcanza en 0 con un valor de la distancia de 811.

Para este ejemplo de 12 nodos el punto Ideal y el punto de Nadir, están dados por

Punto Ideal $= (445, 0)$

Punto Nadir$= (590, 33)$

En las figuras 5.2 y 5.3 se presentan estos resultados gráficamente.

Nótese el compromiso que existe entre las soluciones obtenidas, al disminuir uno de los objetivos se aumenta en el otro y viceversa; por ejemplo, en el caso de 12 nodos dos de los puntos del frente de Pareto que se obtienen son

$$(522, 11) \quad \text{y} \quad (532, 8)$$

se observa en este caso particular que mientras el almacenamiento disminuye tres unidades, la distancia aumenta 10. Esto es, si queremos disminuir en un objetivo,

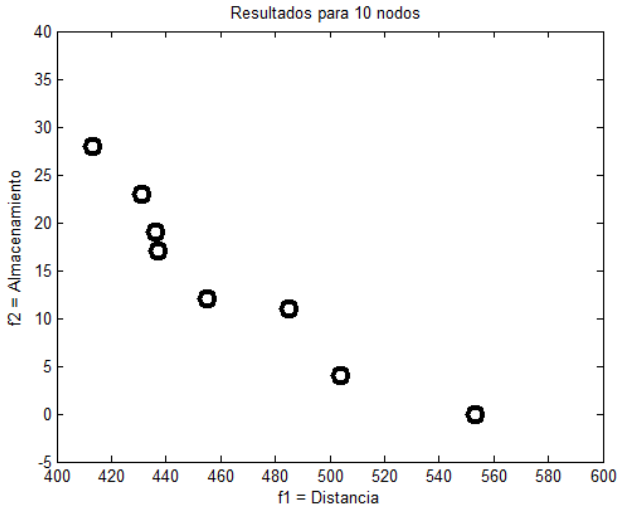


Figura 5.2: Frente de Pareto obtenido para la instancia de 10 nodos

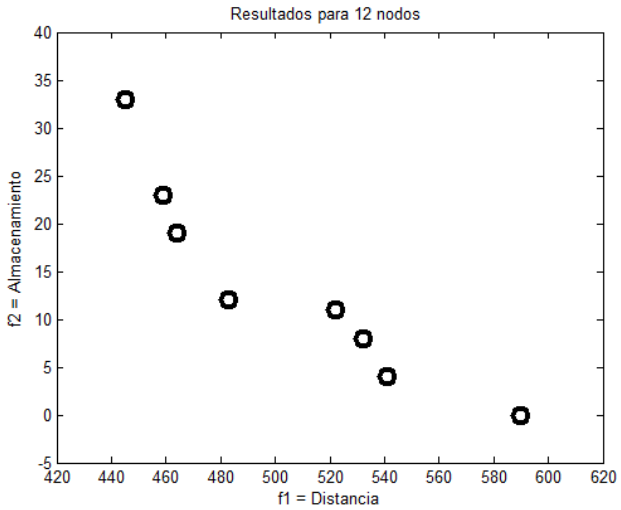


Figura 5.3: Frente de Pareto obtenido para la instancia de 12 nodos

digamos el almacenamiento, entonces tendremos que sacrificar en el otro objetivo, la distancia.

El método de la ε-restricción aproxima bien la frontera de Pareto, sin embargo, su aplicación en nuestro caso con dos objetivos requiere la resolución de p^2 problemas de ruteo mono-objetivo, donde p es el número de valores diferentes que damos al parámetro h. Como hemos mencionado anteriormente, el problema con un solo objetivo es NP-completo, por lo que el uso de este método está limitado a instancias con pocos nodos y para problemas reales de gran tamaño no es adecuada su aplicación.

Método de sumas ponderadas

El método de sumas ponderadas es otro de los métodos tradicionales para problemas multi-objetivo, es uno de los métodos escalares más utilizados; consiste en asociar a cada función objetivo un peso ó ponderación y considerar la optimización de la suma de estas ponderaciones, para dar lugar a un problema mono-objetivo.

En el caso de dos objetivos tendremos que resolver

$$\min_{\mathcal{X}} \lambda_1 f_1(S) + \lambda_2 f_2(S)$$

usualmente se considera que $\lambda_1 + \lambda_2 = 1$. Para aplicar este método se debe tener en cuenta que como los objetivos se ponderan y se suman, éstos deben estar medidos de la misma forma. Se deben escalarizar los valores de los objetivos, estar de preferencia normalizados para poder compararse.

Si bien este método ha sido utilizado ampliamente en la práctica, está documentado [31] que no es adecuado su uso en problemas combinatorios ya que no aproxima de manera eficiente la frontera de Pareto en este tipo de problemas. En el trabajo de Kim y Weck [59] se propone una técnica adaptativa para superar este problema, llamada sumas ponderadas adaptativas. La idea es identificar la región donde es necesario refinar la búsqueda, en estas regiones específicas se diseña

una región factible para un problema de suboptimización, añadiendo restricciones de desigualdad en el espacio de objetivos, en estas regiones se aplican las sumas ponderadas estándar.

Ambos métodos (el clásico de las sumas ponderadas y la adaptación propuesta por Kim y Weck) fueron implementados y ejecutados en las mismas instancias que fueron resueltas con el método de la ε-restricción, pero los resultados no fueron satisfactorios. Sólo se logra encontrar los puntos extremos de la frontera de Pareto y en el caso de 10 órdenes se obtienen, además, dos puntos intermedios.

Por una parte este método no es adecuado en problemas como el nuestro donde el conjunto de soluciones factibles no es un conjunto convexo y por otra parte, al igual que en el método de la ε–restricción, su implementación computacional requeriría la resolución de p^2 problemas mono-objetivo, donde ahora p es el número de sistemas diferentes de pesos considerados. Esto sugiere la necesidad de diseñar e implementar técnicas metaheurísticas multi-objetivo como la que se describen en el siguiente capítulo.

Capítulo 6

Metodologías de solución para el problema biobjetivo

En este capítulo describiremos las metodologías que se proponen para resolver el problema considerando tanto los costos de transportación como los de almacenamiento. En el capítulo anterior, vimos las dificultades inherentes a la aplicación de los métodos clásicos de optimización multi-objetivo a este problema, por lo que nos enfocaremos a desarrollar algoritmos de solución basados en técnicas metaheurísticas de optimización multi-objetivo.

Se presentan dos metodologías. La primera, descrita en la sección 6.1, está basada principalmente en el método MOAMP desarrollados por Caballero et al. [18] acoplado con una búsqueda por entornos variables, adaptada a partir del algoritmo BEV-RF desarrollado en el capítulo 4. La segunda metodología que se propone, descrita en la sección 6.2, está basada en el algoritmo NSGA II, uno de los algoritmos evolutivos más eficientes para problemas multi-objetivo reportados en la literatura.

Recordemos que el problema que queremos resolver, como está planteado en el capítulo 5, consiste en minimizar dos objetivos: el costo de transportación y el de almacenamiento, esto es,

$$\min_{x \in \mathcal{X}} F(x) \qquad \text{donde} \qquad F = (f_1, f_2)$$

$f_1(S) = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$ es el costo total de la distancia recorrida por todos los vehículos y

$f_2(S) = \sum_{i \in V} \sum_{d \in HT} (e_i - d) q_i y_{id}$ es el total de productos almacenados.

Como se mostró en el capítulo 5 los dos objetivos se encuentran en conflicto, esto es, al minimizar sólo la distancia aumenta el almacenamiento y viceversa, por lo que trataremos de encontrar soluciones eficientes al problema, soluciones que no mejoren uno de los objetivos sin demeritar el otro. Aproximaremos el frente de Pareto identificando las soluciones eficientes.

Iniciamos describiendo la primera propuesta de solución basada en el algoritmo MOAMP.

6.1 Descripción del algoritmo de solución basado en MOAMP acoplado con búsquedas por entornos variables biobjetivo: MO-BEV

La primera metodología de solución está basada principalmente en el método MOAMP (Multiobjective Metaheuristic using an Adaptive Memory Procedure) desarrollado por Caballero et al. [18]. Este procedimiento básicamente genera una aproximación a la curva de eficiencia enlazando una serie de procedimientos de búsqueda tabú y realizando posteriormente un proceso de intensificación, que consiste en una exploración del entorno de las soluciones obtenidas hasta ese momento. MOAMP se basa en las siguientes ideas:

- El principio de proximidad de puntos eficientes, según el cual en un entorno o vecindario de una solución eficiente se puede encontrar otra solución eficiente.
- La solución que minimiza la distancia L_∞ al llamado punto ideal, es también eficiente.

Tomaremos como base el esquema general de este método, con las siguientes diferencias:

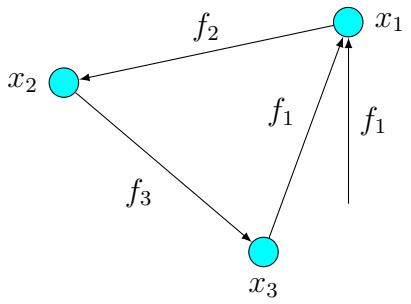


Figura 6.1: Búsquedas enlazadas iniciales

1. En lugar de utilizar búsqueda tabú para las optimizaciones mono-objetivo enlazadas, como se propone en [18], utilizaremos una búsqueda por entornos variables. Esta elección se basa en los buenos resultados que obtuvimos con esta técnica, descrita a detalle en la sección 4.2, al resolver la versión mono-objetivo del problema.

2. En la primera fase del MOAMP [18] se optimiza cada función por separado, tratando de encontrar puntos eficientes cercanos a los extremos de la frontera de Pareto. En un problema con p objetivos se parte de una solución inicial aleatoria y se optimiza sólo f_1, con el punto final obtenido de esta optimización (generalmente un óptimo local) se inicia la optimización de f_2. Al terminar esta búsqueda, se inicia la optimización para f_3 a partir del punto obtenido y así sucesivamente hasta realizar $p+1$ búsquedas. Esto se puede representar con la gráfica que se muestra en la figura 6.1

   En el algoritmo que proponemos en este trabajo, primeramente se construye una solución cercana a la óptima para el objetivo f_1 y ésta se considera solución inicial para el proceso de optimización de f_2. Enseguida se construye una solución cercana a la óptima de f_2 y con ésta se inicia la optimización de f_1. Esto fue motivado por experimentos computacionales que nos indicaban que de esta forma se obtienen más puntos intermedios en la curva inicial de eficiencia.

Entonces en el algoritmo primero se tratará de encontrar puntos eficientes cercanos a los extremos de la frontera de Pareto, minimizando cada función objetivo por separado. Se realizarán una serie de búsquedas por entornos variables minimizando sólo f_1 o sólo f_2, las cuales llamaremos búsquedas de primer tipo. Como se indicó anteriormente, el punto inicial de cada búsqueda será una buena solución obtenida de acuerdo al otro objetivo, esto es, optimizamos f_1 partiendo de una buena solución para f_2 y viceversa. Con esto se intenta encontrar puntos eficientes que aproximen lo mejor posible los óptimos individuales de cada uno de los objetivos.

Todos los puntos visitados durante estas búsquedas se evalúan para verificar si son puntos eficientes, por lo que al terminar estas búsquedas de primer tipo, no solamente tendremos aproximaciones a los extremos de la curva de eficiencia, también tendremos posibles puntos eficientes intermedios, esto es, que se encuentren en los trayectos desde un punto óptimo hasta el siguiente.

Con el propósito de localizar puntos adicionales en el interior de la curva de eficiencia, se utilizará una técnica de la programación compromiso que minimiza la distancia L_∞ al llamado punto ideal, realizando una serie de búsquedas por entornos variables, que llamaremos de segundo tipo. Para esto definimos

$f_i^{\text{mín}}$ y $f_i^{\text{máx}}$: Respectivamente mínimo y máximo valores encontrados para $f_i(S)$ en el conjunto de soluciones no dominadas encontradas, $i = 1, 2$

La función que guiará ahora la búsqueda se define como

$$F_\lambda(S) = \text{máx}\left\{\lambda\frac{f_1(S) - f_1^{\text{mín}}}{f_1^{\text{máx}} - f_1^{\text{mín}}}, (1-\lambda)\frac{f_2(S) - f_2^{\text{mín}}}{f_2^{\text{máx}} - f_2^{\text{mín}}}\right\} \qquad \text{donde} \;\; \lambda \in (0,1)$$

Recordemos que una de las premisas del MOAMP es que si un punto minimiza la distancia L_∞ normalizada al punto ideal, entonces es un punto eficiente. Obsérvese que precisamente la función F_λ representa esta distancia ponderada por λ y $(1-\lambda)$. Se ha observado que las soluciones que minimizan F_λ son también eficientes.

En general, un punto que minimiza una distancia $L_p, p \in [1, \infty]$ normalizada al punto ideal es también un punto eficiente. Al conjunto de todos los puntos eficientes

obtenidos de esta forma se le denomina Conjunto Compromiso [81]. Estos puntos tienen la característica común de representar un buen equilibrio entre los criterios, es decir, son puntos que sin ser muy buenos en algún criterio particular presentan buenos valores para todos ellos. En especial, el punto que minimiza la distancia L_∞ suele representar el mejor equilibrio posible entre los distintos criterios.

Lo que se pretende al tratar de encontrar puntos del conjunto compromiso, es completar una muestra de puntos eficientes suficientemente diversa para posteriormente, intensificar la búsqueda sobre estos puntos y obtener una muestra lo más amplia posible del conjunto eficiente.

Para todos los puntos visitados en cada una de las búsquedas, del primer o segundo tipo, se comprueba si está ó no dominado por los puntos eficientes encontrados hasta el momento. Si no está dominado se envía al conjunto de puntos eficientes y éste se actualiza de ser necesario. En otras palabras, se comprueba en primer lugar si el punto está dominado por alguno de los elementos ya existentes en la lista de puntos eficientes. De ser así, se desecha. En otro caso, el punto se incluye en la lista de eficientes y seguidamente se eliminan de esta lista aquellos puntos que estén dominados por este nuevo elemento.

Con esto se obtiene una lista de todos los puntos eficientes que han sido visitados mediante cada una de las búsquedas, las cuales esperamos que exploren distintas zonas del conjunto eficiente. Así, enlazando las distintas búsquedas se pretende realizar una exploración de este conjunto. Al final obtendremos una muestra de la curva de eficiencia distribuida tanto por las zonas en los que uno de los criterios es predominante como por aquellas zonas caracterizadas por un equilibrio entre los distintos criterios.

Una vez que se obtiene esta muestra de puntos eficientes, se intensifica la búsqueda sobre cada uno de ellos explorando sus vecinos buscando nuevos puntos eficientes. Estos vecinos serán generados a partir de cuatro movimientos básicos que serán explicados más adelante.

En el siguiente apartado se presentan los detalles del algoritmo propuesto.

6.1.1 Algoritmo MO-BEV

Con los elementos ya descritos formulamos el primer algoritmo que se propone para generar la aproximación a la curva de eficiencia, el cual consiste en búsquedas por entornos variables enlazadas, dentro de la estructura del MOAMP. El algoritmo consta de tres fases: la primera y segunda fase realizan las búsquedas del primer y segundo tipo, respectivamente; la tercera fase es la intensificación.

Denotaremos por `Set_ND` el conjunto de soluciones no dominadas encontradas durante la búsqueda. El conjunto `Set_ND` final será la aproximación obtenida a la curva de eficiencia en el espacio de decisión.

Fase I: Aproximación a los extremos de la curva de eficiencia

Hacer `Set_ND` $= \emptyset$

Ejecutar `Constructivo_Distancia` para obtener una buena solución inicial S considerando $f_1(S)$ como función objetivo

Hacer `Set_ND` $= \{S\}$

Ejecutar `BEV-RFB`(S) considerado $f_2(S)$ como función objetivo

Ejecutar `Constructivo_Stock` para obtener una buena solución inicial S considerando $f_2(S)$ como función objetivo

Ejecutar `BEV-RFB`(S) considerado $f_1(S)$ como función objetivo

Fase II: Búsqueda de soluciones compromiso

Hacer niter=0, `iter_ultimo_cambio` =0

Repetir

niter = niter +1

Generar $\lambda \in (0, 1)$ aleatoriamente

Ejecutar `BEV-RFB`(S) usando $F_\lambda(S)$ como función objetivo

Si en esta ejecución `Set_ND` ha cambiado hacer

`iter_ultimo_cambio` = niter

Hasta niter > `iter_ultimo_cambio` + `max_iter`

Fase III: Intensificación

$\forall S \in$ `Set_ND` marcar S como "no-explorado"

Repetir

$\forall S \in \{$ `Set_ND` $/S\}$ marcado como "no-explorado"hacer:

Marcar S como "explorado"

Ejecutar `Exploracion_vecinos`(S)

Marcar como "no-explorado" todas las soluciones

introducidas en `Set_ND` tras la última exploración

Hasta que `Set_ND` se estabilice (no cambie)

Como se explicó anteriormente, la primera fase pretende obtener las mejores soluciones con respecto a cada una de las funciones objetivos, mientras que la segunda fase busca encontrar una muestra diversa de soluciones "compromiso". Esta segunda fase finaliza cuando trascurre un cierto número de iteraciones (`max_iter`) sin cambio en `Set_ND`. Hay que indicar que dentro de cada ejecución de BEV-RFB se comprueba si cada solución visitada se incorpora a `Set_ND`. En otras palabras, el conjunto de soluciones no dominadas se actualiza con todas las soluciones visitadas durante cada ejecución del procedimiento BEV-RFB en estas dos primeras fases. Finalmente, en la tercera fase, se completa el conjunto de soluciones no dominadas explorando el vecindario de las soluciones no dominadas encontradas hasta ese momento.

Describimos enseguida a detalle cada uno de los procedimientos que intervienen en el algoritmo.

Procedimiento `Constructivo_Distancia`

Con el procedimiento `Constructivo_Distancia` se construye una buena solución considerando solamente la distancia total recorrida, esto es, tomando en cuenta solamente la función objetivo $f_1(S)$. Para ello se ejecuta el procedimiento híbrido de dos fases, GRASP con encadenamiento de trayectorias (HGET) explicado en capítulo 4, el cual construye una buena solución considerando como objetivo la distancia total recorrida.

Procedimiento `Constructivo_Stock`

Con el procedimiento `Constructivo_Stock` se construye una buena solución considerando como función objetivo el almacenamiento o stock $f_2(S)$. Teniendo en cuenta que siempre se puede obtener un almacenamiento con valor cero si los pedidos se entregan el día de su correspondiente deadline, la estrategia consiste en agrupar los pedidos según su correspondiente deadline y diseñar las rutas de reparto para cada grupo ejecutando HGET con los mismos parámetros que en `Constructivo_Distancia`. De esta manera la solución obtenida tiene el mínimo almacenamiento (0), ya que los pedidos son entregados en su correspondiente *deadline* con una distancia total aceptable, esto es, al menos cercana a la distancia mínima para cada conjunto de días.

Procedimiento BEV-RFB

El procedimiento BEV-RFB se ejecuta tanto considerando las funciones objetivos iniciales $f_1(S)$ y $f_2(S)$ como las diferentes funciones mixtas $F_\lambda(S)$. Es fácil observar que $f_1(S)$ y $f_2(S)$ pueden ser consideradas un caso particular de $F_\lambda(S)$ si se toma λ igual a 1 ó 0, respectivamente. El procedimiento se puede describir de la forma siguiente.

Procedimiento BEV-RFB(S)

niter = 0

Repetir

  niter = niter+1

  $r := 0$

  Repetir

    $r := r + 1$

    Obtener S_1 = **Agitación** (r, S)

    Mejorar S_1 ejecutando `Busqueda_Local_Mixta` (S_1)

    Actualizar `Set_ND` con S

  hasta $(r = rmax)$ o $(F_\lambda(S_1) < F_\lambda(S))$

  Si $F_\lambda(S_1) < F\lambda(S)$ entonces hacer $S := S_1$

Hasta satisfacer un criterio de parada

La actualización de `Set_ND` consiste en que si S no está dominada por ninguna solución en `Set_ND`, añadir S a `Set_ND` y eliminar aquellas soluciones de `Set_ND` dominadas por ella.

Como criterios de parada se pueden considerar: número máximo de iteraciones sin mejorar la solución actual S, tiempo máximo de ejecución, etc. En nuestro caso optamos por un número máximo de iteraciones sin mejora.

El proceso de **Agitación** es el mismo que se utiliza en BEV-RF (el algoritmo de solución basado en búsqueda por entornos variables para el problema mono-objetivo) y está descrito a detalle en la sección 4.2.1 No obstante lo recordamos brevemente enseguida y describimos a continuación la `Busqueda_Local_Mixta`.

Con el procedimiento **Agitación** se genera una solución que pertenece a una vecindad de la solución actual S. Esta vecindad depende del parámetro r, de forma que valores bajos de r se corresponden con entornos pequeños, es decir genera

soluciones cercanas o parecidas a S; mientras que a valores altos de r corresponden entornos mayores, es decir soluciones más alejadas de S. El procedimiento está basado en la asociación de soluciones a grandes cadenas. Las grandes cadenas se dividen aleatoriamente en r subcadenas, las cuales se permutan también de forma aleatoria y la nueva cadena se divide en rutas factibles resolviendo un problema de la ruta más corta sobre un grafo auxiliar convenientemente definido.

Las soluciones obtenidas por el procedimiento de **Agitación** se mejoran utilizando el procedimiento `Busqueda_Local_Mixta`, una búsqueda local que considera los dos objetivos. Este es un método iterativo que en cada iteración se mueve de la solución actual a una mejor solución vecina. El procedimiento finaliza cuando no se encuentran mejores soluciones vecinas.

Si denotamos por $N(S)$ el vecindario de una solución S, el procedimiento de `Busqueda_Local_Mixta` puede ser descrito como sigue.

Procedimiento `Busqueda_Local_Mixta`(S)

Repetir

  `valor_anterior`$= F_\lambda(S)$

  Buscar $F_\lambda(S^*) = \min\{F_\lambda(S')/S' \in N(S)\}$

  Si $F_\lambda(S^*) < F_\lambda(S)$ hacer $S = S^*$

Hasta que $F_\lambda(S^*) \geq$ `valor_anterior`

El vecindario de una solución S, esto es, el conjunto $N(S)$ de soluciones vecinas, se obtiene con cambios simples en S. Más específicamente, consideramos 4 cambios para obtener una solución vecina, los tres primeros fueron utilizados en la búsqueda local del problema mono-objetivo y el cuarto movimiento se añadió específicamente para el caso biobjetivo.

a) Intercambio de dos cadenas de vértices consecutivos en una misma ruta, movi-

miento tipo Or.

b) Intercambio de dos cadenas de vértices en dos rutas diferentes, movimientos tipo Taillard.

c) Mover una cadena de vértices de una ruta a otra ruta diferente.

d) Intercambio de dos cadenas de vértices no consecutivos en una misma ruta.

Teniendo en cuenta los resultados de los experimentos computacionales realizados para el problema mono-objetivo (descritos en el siguiente capítulo) utilizamos el mecanismo de la búsqueda local rápida para acelerar este procedimiento, descrita detalladamente en 4.1.2. En el caso que nos ocupa ahora se definen cuatro tipos de subvecindarios para cada solución S.

$A(j)$ =Soluciones que resultan de intercambiar dos cadenas de vértices consecutivas en una ruta j, $j = 1, \ldots, K$

$B(j, j')$ = Soluciones que resultan de intercambiar cadenas de vértices que pertenecen a las rutas j y j', $j = 1, \ldots, K-1$ y $j' = j+1, \ldots, K$

$C(j, j')$ =Soluciones que resultan de mover una cadena de vértices de la ruta j a la ruta j', $j, j' = 1, \ldots, K$ y $j' \neq j$

$D(j)$ = Soluciones que resultan de intercambiar dos cadenas de vértices en una ruta j, $j = 1, \ldots, K$

Con ese esquema se logra agilizar el proceso de búsqueda sin demeritar la calidad en la solución.

Procedimiento `Exploracion_vecinos`

El procedimiento `Exploracion_vecinos` que se usa en la Fase III del algoritmo MO- BEV consiste en explorar los vecinos de las soluciones encontradas hasta el momento buscando nuevas soluciones no dominadas. Para cada solución S se considera

su vecindario $N(S)$ como se describió en el procedimiento de `Busqueda_Local_Mixta`, soluciones que pueden obtenerse a partir de S considerando los cuatro movimientos básicos. Para cada una de estas soluciones vecinas se verifica si pertenece al conjunto de soluciones no dominadas, de ser así se actualiza este conjunto eliminando las soluciones que estaban en él y que son dominadas por la nueva solución. El procedimiento puede ser descrito de la forma siguiente:

Procedimiento `Exploracion_vecinos`

$\forall S' \in N(S)$

Si S' no está dominada por ninguna solución en `Set_ND` entonces

Introducir S' en `Set_ND`

Eliminar las soluciones en `Set_ND` dominadas por S'

Con esto queda completa la descripción del algoritmo que se propone para resolver el problema biobjetivo: MO-BEV.

Dado que en la literatura científica no existen algoritmos específicos para el problema planteado que nos permita realizar una comparación con el algoritmo propuesto, decidimos desarrollar otro método de solución basado en la filosofía de la metaheurística NSGA-II, para posteriormente comparar con MO-BEV. Lo presentamos a continuación.

6.2 Descripción del algoritmo de solución basado en NSGA-II: NSGA-RF

La metaheurística NSGA-II desarrollada por Debb et al. [27], es un algoritmo genético elitista que ha demostrado obtener buenos resultados en muchos problemas multi-objetivo en los últimos años. Por ello, consideramos conveniente desarrollar

una metodología basada en ella y posteriormente compara ambas metodologías.

Tomando la estructura básica de este algoritmo, diseñamos operadores adecuados a nuestro problema para aproximar la curva de eficiencia. Primeramente describimos la estructura básica del NSGA-II.

Sin entrar en detalles, NSGA-II es un algoritmo genético que opera sobre un conjunto de soluciones, una población de soluciones, y usa operadores típicos de estos algoritmos: Selección de padres, Cruce, Mutación, Renovación o Selección de individuos que pasan a la siguiente generación, etc. Por otra parte, dado un conjunto P de soluciones, éste se puede descomponer en capas de no-dominancia, $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3, \ldots$ que se definen de la siguiente manera:

$\mathcal{F}_1$: Conjunto de soluciones no dominadas de P

$\mathcal{F}_2$: Conjunto de soluciones no dominadas de $P \backslash \mathcal{F}_1$

$\mathcal{F}_3$: Conjunto de soluciones no dominadas de $P \backslash \{\mathcal{F}_1 \cup \mathcal{F}_2\}$

y así sucesivamente.

Asimismo, para cada solución S se definen los siguientes valores:

- $ind(S)$ índice de la capa de no-dominancia a la que pertenece S
- `cwd_dst` (S) (de "crowding distance" en inglés): medida de la dispersión de los puntos no dominados. En nuestro caso se define como

$$\texttt{cwd\_dst}(S) = \frac{f_1(S_a) - f_1(S_b)}{f_1^{\text{máx}} - f_1^{\text{mín}}} + \frac{f_2(S_c) - f_2(S_d)}{f_2^{\text{máx}} - f_2^{\text{mín}}}$$

  donde S_a y S_b son las soluciones justo anterior y posterior a S al ordenar su capa de no-dominancia ($\mathcal{F}_{ind}(S)$) según los valores de f_1 en orden descendente. Similarmente, S_c y S_d son las soluciones justo anterior y posterior a S al ordenar $\mathcal{F}_{ind}(S)$ según los valores de f_2 en orden descendente. En caso de que S tome valores extremos dentro de su capa de no-dominancia, bien para f_1 o bien para f_2, entonces `cwd_dst` $(S) = \infty$.

El valor `cwd_dst` mide en cierta forma la dispersión de los puntos no dominados aproximando el perímetro del cuadrado formado con los puntos no dominados más cercanos al punto en cuestión. Este perímetro se determina calculando la distancia entre las dos soluciones más cercanas al punto a lo largo de los dos objetivos. Estas distancias se normalizan dividiéndolas entre la diferencia entre los valores máximos y mínimos de los correspondientes objetivos. Las distancias ya normalizadas se suman para dar el valor de `cwd_dst` para el punto en cuestión. Para los puntos extremos este valor se considera infinito.

La función principal de este índice `cwd_dst` es poder definir un orden parcial $<_C$, el cual se define de la forma siguiente:

Sean dos soluciones S y S', se dice que $S <_C S'$ siempre que $(ind(S) < ind(S'))$ o $\{(ind(S) = ind(S'))$ y $(\texttt{cwd\_dst}\ (S) > \texttt{cwd\_dst}\ (S'))\}$

Esto es, entre dos soluciones en diferentes frentes preferimos aquella con menor valor del índice en la capa de no-dominancia. De otra forma, si ambas soluciones pertenecen al mismo frente, preferimos la solución que tenga un valor mayor de `cwd_dst`, esto es, la que esté más aislada.

Este operador se utiliza en la selección de padres y en la selección de las soluciones que sobreviven en la siguiente generación. Con esto se trata de favorecer en primer lugar la selección de soluciones menos dominadas, y en segundo lugar de soluciones más aisladas dentro de su capa de dominancia para tratar de rellenar esa región.

Con estos elementos presentamos enseguida el algoritmo evolutivo basado en NSGA-II, dotado de mecanismos de cruce y mutación desarrollados específicamente para el problema. Lo denotamos como NSGA-RF.

6.2.1 Algoritmo NSGA-RF

Considerando n_{pob} el tamaño de la población inicial, p_{mut} la probabilidad de mutación, el esquema general del algoritmo propuesto es el siguiente.

Algoritmo NSGA-RF

1. Generar una población de soluciones iniciales P_0 de tamaño n_{pob}
2. Dividir P_0 en capas de no dominancia
3. Hacer $t := 0$
4. Repetir
   a) Calcular el valor `cwd_dst` para los elementos de P_t
   b) Generar la población Q_t de soluciones hijas de tamaño n_{pob} a partir de P_t
   c) Aplicar el operador mutación a los elementos de Q_t
   d) Hacer $R_t = P_t \cup Q_t$ y dividir R_t en capas de no dominancia
   e) Formar P_{t+1} con los n_{pob} primeros elementos de R_t según el orden parcial $<_C$
   f) Hacer $t := t + 1$

   Hasta alcanzar criterio de parada;

Presentamos enseguida cómo se diseñaron cada uno de los operadores para resolver el problema de minimizar costos de transportación y de almacenamiento.

Generación de la población de soluciones iniciales

Para generar cada una de las soluciones iniciales de P_0 se genera aleatoriamente una permutación con el conjunto de órdenes $\{1, 2, \ldots, n\}$ y se escribe en forma de

gran cadena. Esta gran cadena se divide en subcadenas correspondientes a rutas factibles que constituyen la solución. Esta división se realiza en forma similar a como se explicó en el algoritmo HGET (ver sección 4.1.3) para el caso mono-objetivo.

Operadores de selección y cruce

Para generar el conjunto de soluciones hijos Q_t se utilizan las operaciones de selección de padres y cruce y se procede como sigue:

- Se seleccionan n_{pob} soluciones padre. Cada uno de ellos se selecciona tomando 2 elementos de P_t al azar (con reemplazamiento) y tomando el mejor según el orden $<_C$.
- Con n_{pob} soluciones padre se forman $(n_{pob}/2)$ parejas, y cada una de ellas se cruza siguiendo las ideas de Beasley [12] y Prins [80] que fueron desarrolladas en el capítulo 4 (ver sección 4.1.3). Primeramente se forma una gran cadena a partir de cada solución, enlazando las rutas y eliminando el depósito. A manera de ilustración, supongamos que S_1 se compone de dos rutas:

  $$0-1-4-2-3-0 \quad \text{y} \quad 0-5-6-8-7-0$$

  entonces, la gran cadena asociada BT_1 está dada por: $1-4-2-3-5-6-8-7$.

  Si S_2 se compone de las rutas:

  $$0-1-2-6-8-0 \quad \text{y} \quad 0-5-4-3-7-0$$

  entonces, la gran cadena en este caso es BT_2: $1-2-6-8-5-4-3-7$.

  Seguidamente se generan 2 puntos de cruce i y j para partir cada subcadena en 3 partes: la primera de la posición 1 al $i-1$, la segunda de la posición i a la j, y la tercera de la $j+1$ a la n, como se ilustra a continuación.

  $$BT_1 = \quad 1-4-2 \qquad 3-5-6 \qquad 8-7$$

  $$BT2 = \quad 1-2-6 \qquad 8-5-4 \qquad 3-7$$

Posteriormente se obtienen dos grandes cadenas hijas BH_1 y BH_2 de la siguiente forma: El segundo fragmento de BT_1 se copia en BH_1 en la misma posición; para completar BH_1 se va "barriendo" BT_2 circularmente empezando en la posición $j+1$ copiando en BH_1 los nodos que faltan. BH_2 se obtiene de forma análoga (cambiando los "roles" de BT_1 y BT_2). A continuación se muestran BH_1 y BH_2.

$$BH_1 = \quad 2-8-4-3-5-6-7-1$$

$$BH_2 = \quad 2-3-6-8-5-4-7-1$$

Finalmente las cadenas BH_1 y BH_2 se descomponen en subcadenas correspondientes a rutas factibles que constituyen la soluciones hijas Sh_1 y Sh_2. Esto se realiza en forma similar a lo descrito en la sección 4.1.3.

Operador mutación

Cada solución hija puede mutar con una probabilidad p_{mut}. La mutación, siguiendo la idea de Jozefowiez et al. [54] para problemas de ruteo, consiste en tomar dos rutas al azar así como una cadena de puntos de una de ellas y mover dicha cadena de una ruta a la otra. La posición donde se inserta en la segunda ruta es la mejor en cuanto a la distancia a recorrer, es decir la que menos incrementa la distancia. Esta operación se corresponde con los movimientos tipo c) del procedimiento `Busqueda_Local_Mixta` (ver sección 6.1.1). Note que dentro de una ruta el día de entrega no cambia, sólo la distancia recorrida, pero sí puede cambiar de una ruta a otra. Por ello, para realizar esta operación debe verificarse la factibilidad de este movimiento. Finalmente las rutas afectadas son mejoradas con los movimientos a) y d) del procedimiento `Busqueda_Local_Mixta`.

Proceso de renovación o selección de sobrevivientes para la próxima generación

La división de un determinado conjunto de soluciones (P_0 en el paso 2, o R_t en el 4.d) se realiza mediante el procedimiento *fast-not-dominated-sort* propuesto en

el trabajo de Deb [27]. El proceso de renovación (paso 4.e), esto es, determinar los elementos de R_t que sobreviven en la siguiente generación, se realiza de la forma siguiente:

- Denotemos por $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \ldots$ las capas de no-dominancia en que se divide R_t
- Hacer $i := 1$ y $P_{t+1} = \emptyset$
- Mientras $|P_{t+1}| + |\mathcal{F}_i| \leq n_{pob}$ hacer $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$ y $i := i + 1$
- Calcular el valor de `cwd_dst` para los elementos de $\mathcal{F}_i$
- Incluir en P_{t+1} los $n_{pob} \backslash |P_{t+1}|$ elementos de $\mathcal{F}_i$ con mayor valor de `cwd_dst`.

Finalmente, como criterio de parada se puede tomar: un número máximo del total de iteraciones, número de iteraciones sin cambios en P_t, tiempo máximo de computación, etc. En nuestra implementación, para realizar una comparación justa con los resultados obtenidos con MO-BEV, utilizamos como criterio de paro el tiempo máximo de computación.

Con esto queda completa la descripción de los operadores diseñados para el método de solución basado en NSGA-II para nuestro problema.

Estas son las dos metodologías propuestas, una basada en el algoritmo MOAMP y la segunda en NSGA-II. Los resultados obtenidos en la vasta experimentación computacional realizada con ambos se encuentran en el capítulo 7.

Capítulo 7

Experimentos computacionales

Las metodologías propuestas para abordar el problema tanto desde la perspectiva mono-objetivo como de la biobjetivo fueron implementadas computacionalmente. En este capítulo presentamos algunos resultados de los experimentos realizados.

Los experimentos se realizaron primeramente en instancias de tamaño pequeño generadas aleatoriamente para probar la efectividad de las metodologías y, posteriormente en instancias pseudo–reales con datos similares a los del problema real que motivó este estudio. La descripción de las instancias se presenta en la primera sección.

En la segunda sección se presentan los resultados al abordar el problema mono-objetivo de minimizar solamente la distancia total recorrida, con las dos metodologías propuestas: GRASP con encadenamiento de trayectorias (algoritmo HGET) y búsqueda por entornos variables (algoritmo BEV-RF).

La tercera sección está dedicada a los experimentos computacionales realizados con las metodologías desarrolladas para resolver e problema biobjetivo de minimizar la distancia total recorrida y el almacenamiento simultáneamente.

Los algoritmos fueron implementados en Object Pascal utilizando el compilador Borland Delphi (ver 5.0) y se ejecutaron en Corel 8400 PC a 2.66 GHz CPU.

En las instancias pequeñas se resuelve el problema de forma exacta con software comercial, específicamente se utilizó Cplex v.11.2 en una Sun Fire V440 Ultra Sparc

III, 1062 GHz. Cplex se ejecutó con los parámetros por default, en particular con una tolerancia de error relativo de 10^{-4} para el optimizador de problemas entero mixtos.

7.1 Descripción de las instancias

Para la realización de los experimentos se generaron tres tipos de instancias. Las primeras fueron generadas de forma aleatoria y se utilizaron para comprobar la efectividad de los algoritmos metaheurísticos diseñados en alcanzar el óptimo en instancias con pocas órdenes. Las instancias del segundo y tercer tipo se generaron con datos pseudoreales y se utilizaron para calibrar parámetros y evaluar el desempeño de procedimientos específicos dentro de los algoritmos metaheurísticos. El tercer grupo de instancias, además, fue utilizado para comparar resultados con los obtenidos por otros algoritmos.

En los tres tipos de instancias se considera un horizonte de planeación de 5 días, de lunes a viernes, y los *deadlines* de las órdenes se generan de forma aleatoria entre 1 y 5.

Las instancias del primer tipo son las mismas que se utilizaron para validar las formulaciones matemáticas del modelo mono-objetivo, éstas fueron descritas en el capítulo 3. Las localizaciones fueron generadas aleatoriamente en el plano cartesiano en la región $[-25, 25] \times [-25, 25]$; el depósito se ubica en (0,0). Se consideran un número pequeño de órdenes, $n \in \{10, 12, 15\}$. La matriz de costos se forma con la distancia euclideana redondeada; las demandas se generaron entre 1 y 15 para cada orden y la capacidad de los vehículos se fijó en 30. Se consideran varios valores de g, el número máximo de días permitidos para adelantar pedidos, desde 0 hasta 4, es decir, desde un ruteo día a día hasta el máximo posible. El número de vehículos disponibles $K \in \{2, 3, 4\}$.

Las instancias del segundo tipo se generaron considerando como localizaciones

algunas ciudades de España a donde se envían productos desde la fábrica central. Los datos son de la siguiente manera:

- Las localizaciones se sitúan geográficamente en diferentes poblaciones del norte de España, concretamente el depósito central se ubica en Briviesca, Burgos, donde la empresa tiene su fábrica.
- Las distancias entre localizaciones son las distancias por carreteras.
- La capacidad de los vehículos se fija en 30.
- La demanda de cada pedido se genera entre 1 y $Q_{\text{máx}}$, donde $Q_{\text{máx}} \in \{15, 22\}$.
- Se toman 41 ciudades a las que se envían los pedidos, si todas las ciudades requieren sólo un pedido por semana se considera que se trabaja con un grafo de 42 nodos al contar del depósito. Se considera también que algunas ciudades requieran dos o más pedidos por semana, por lo que el número de nodos varía alrededor de 100 a 190 pedidos por semana, que corresponden a que algunas ciudades requieren hasta cuatro pedidos por semana, algunas tres, dos o sólo uno.
- Para cada combinación de parámetros $Q_{\text{máx}}$ y n se generan 5 instancias y para cada instancia se consideran varios valores de g, el número máximo de días permitidos para adelantar pedidos. En las instancias más pequeñas desde $g = 0$ (ruteo día a día) hasta $g = 4$; en las instancias más grandes sólo $g = 0$ y $g = 1$ (adelanto máximo de un día).

Las instancias del tercer tipo son similares a las del segundo tipo con los siguientes cambios:

- $Q_{\text{máx}} = 12$
- $n \in \{42, 62, 82, 102\}$.
- Para cada valor de n se generan 5 instancias considerando solamente $g = 1$.

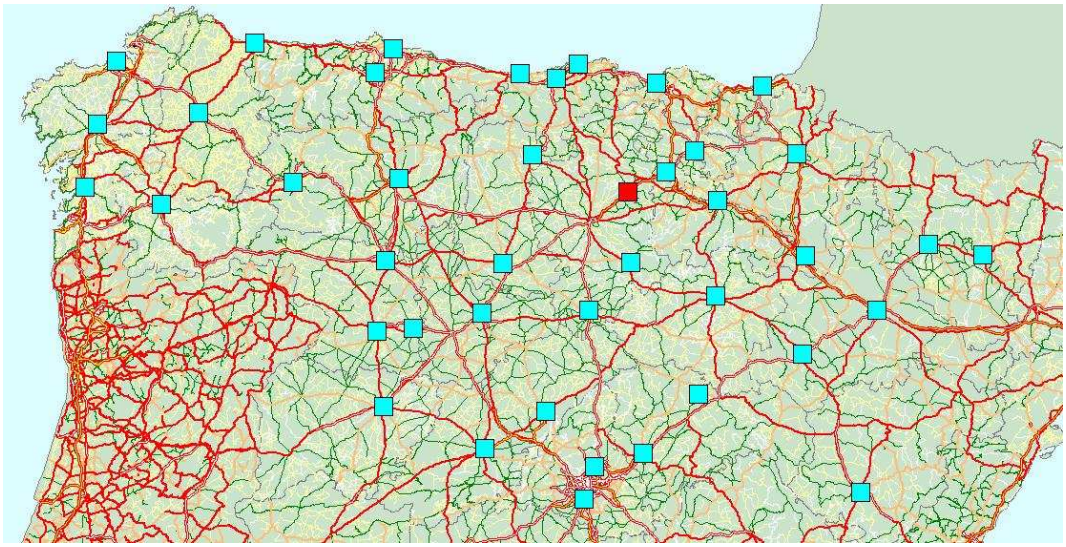


Figura 7.1: Centros de distribución. Depósito en rojo

- A cada orden se le asignan las correspondientes coordenadas (x, y) de su centro de distribución. En este caso las distancias se obtienen como las distancias euclideanas, no como las distancias por carretera. La motivación para esto es que estas instancias son las que se utilizan para comparar los resultados de nuestros algoritmos con los algoritmos publicados en literatura para el PVRP (VRP Periódico) y en la mayoría de los reportes las instancias en la literatura utilizan este formato para los datos.

7.2 Resultados computacionales para el modelo mono-objetivo

En el capítulo 4 se describieron las metodologías que se proponen para el problema con un solo objetivo: Híbrido de GRASP con encadenamiento de trayectorias (HGET) y una búsqueda por entornos variables (BEV-RF). Presentamos enseguida algunos resultados de experimentos computacionales realizados para comprobar la efectividad de estos métodos.

Primeramente en la sección 7.2.1 se presentan los resultados obtenidos utili-

zando HGET. Enseguida en el apartado 7.2.2 se presentan los resultados con BEV-RF y finalmente en la sección 7.2.3, una comparación de los resultados entre las dos metodologías y otros algoritmos para el PVRP.

Recuérdese que en el problema mono-objetivo se mide solamente la distancia total recorrida.

7.2.1 Experimentos para evaluar el desempeño del algoritmo HGET

La primera metodología que se propuso, descrita a detalle en el capítulo 4, fue un algoritmo híbrido de dos fases, la primera fase es un GRASP y la segunda consiste en un encadenamiento de trayectorias. A continuación se presentan los resultados obtenidos al probar este algoritmo. Primero se realizan experimentos solamente con la Fase I, esto es, sólo con el GRASP y después se incorpora la Fase II para medir el efecto del encadenamiento de trayectorias en la calidad de las soluciones obtenidas.

Experimentos para evaluar los procedimientos diseñados en el GRASP

Se realizaron experimentos para probar la fase I del algoritmo HGET, primero verificando que logra alcanzar el óptimo en instancias pequeñas y después se evalúa su desempeño en en instancias grandes. Dentro de estos experimentos se hace una evaluación de la búsqueda local y otros procedimientos y se calibran parámetros.

Validación del algoritmo. Comparación con Cplex

Los primeros experimentos realizados fueron para comprobar que el GRASP es capaz de obtener la solución exacta en instancias pequeñas, donde fue posible obtener la solución óptima en un tiempo razonable con software comercial. Como ya se comentó anteriormente se utilizaron instancias del primer tipo para comparar con

los resultados que se obtuvieron con estas mismas instancias en el capítulo 3 con el modelo II.

En estos ejemplos se realizaron 1000 iteraciones y se consideraron los valores de $\alpha = 0.5, \beta = 1$, Los resultados se muestran en la tabla 7.1.

En todos los resultados que se presentan, el tiempo se reporta en segundos y el costo mide la distancia total recorrida por los vehículos en todo el horizonte de planeación. En este experimento en todas las instancias se alcanza la solución óptima de acuerdo a los resultados reportados en el capítulo 3, por lo que aqui sólo reportamos el tiempo que necesitan tanto Cplex como el GRASP en obtener la solución.

Tabla 7.1: Resultados GRASP: Tiempos en instancias aleatorias

	$n = 10$		$n = 12$		$n = 15$	
g	CPLEX	GRASP	CPLEX	GRASP	CPLEX	GRASP
1	3.42	0.641	11	0.763	284	1.148
2	8.99	0.672	160	1.076	53278	1.565
3	54.95	0.834	1658	1.248	110621	1.939
4	79.49	1.117	10295	1.868	615973	2.601

Se puede observar que el GRASP logra llegar a la solución óptima en un tiempo de cómputo muy corto en estas instancias pequeñas, comparado con el tiempo que requiere Cplex para llegar a la misma solución Estas pruebas fueron realizadas para validar que el software comercial tiene dificultades para resolver aún estas instancias pequeñas, mientras que el GRASP requiere pocos segundos para llegar a la solución óptima.

Estos resultados indican que es apropiado aplicar este algoritmo para resolver instancias mas grandes donde desconocemos la solución óptima.

Evaluación de la búsqueda local

Con el objetivo de investigar qué tanto era capaz la búsqueda local de mejorar la calidad de la solución construida en la fase de construcción del GRASP, se realizaron los experimentos que describimos a continuación.

En esta parte se utilizaron las instancias del segundo tipo. Primeramente se evalúa la efectividad de la propuesta de la búsqueda local rápida, la cual consiste en agilizar la búsqueda local al subdividir los vencindarios asociados a cada solución, los detalles de la descripción se documentan en 4.1.2. Para ello se resolvieron las instancias utilizando el GRASP (fase I del HGET) con la búsqueda local completa y con la búsqueda local rápida comparando resultados.

En estos experimentos se realizaron 1000 iteraciones y se considera $\alpha = \beta = 0$ para evaluar la fase de búsqueda local. Algunos de los resultados se muestran en la tabla 7.2 y la tabla completa en B.1 en el apéndice B.

Se puede observar que la calidad de los resultados es la misma, se obtienen los mismos resultados en el valor de la función objetivo, pero los tiempos necesitados para encontrar la solución bajan considerablemente al utilizar la búsqueda local rápida, por lo que consideramos muy adecuado utilizar esta estrategia que consume menos tiempo de cómputo con la misma calidad en la solución obtenida.

Con el objetivo de reducir el tiempo de cómputo en los experimentos que siguen se utilizará siempre la búsqueda local rápida.

Veamos ahora el efecto de la búsqueda local en la calidad de la solución obtenida. Mostramos enseguida los costos y tiempos consumidos para las mismas instancias aplicando solamente el procedimiento de construcción del GRASP, esto es, sin búsqueda local. Algunos de los resultados se muestran en la tabla 7.3, los resultados completos se encuentran en la tabla B.2 en el apéndice B; se realizaron 1000 iteraciones.

Se observa que la búsqueda local sí proporciona un beneficio al ser aplicada,

Tabla 7.2: Resultados GRASP: Comparación de resultados con búsqueda local completa y búsqueda local rápida

				Búsqueda Local Completa			Búsqueda Local Rápida		
n	$Q_{\text{máx}}$	Núm.de Instancia	g	costo	Tiempo Total	Tiempo mejor sol.	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.591	0.090	111414	0.240	0.050
42	15	1	1	90934	3.625	3.475	90934	1.582	1.512
42	15	1	2	88874	4.867	2.043	88874	2.264	0.992
42	15	1	3	82973	10.565	0.871	82973	4.236	0.370
42	15	1	4	80473	17.595	2.874	80473	7.211	1.272
42	15	2	0	112218	1.181	0.250	112218	0.501	0.121
42	15	2	1	93439	7.020	5.678	93439	2.924	2.414
42	15	2	2	86529	16.204	2.063	86529	6.589	1.011
42	15	2	3	82344	33.238	9.884	82344	13.730	4.647
42	15	2	4	80011	42.691	0.480	80011	17.846	0.250
42	15	3	0	111547	0.841	0.180	111547	0.370	0.090
42	15	3	1	90763	4.457	0.091	90763	1.913	0.050
42	15	3	2	84838	7.711	0.240	84838	3.324	0.160
42	15	3	3	81690	13.620	2.895	81690	5.238	1.172
42	15	3	4	78942	13.589	8.071	78942	5.658	3.495

Tabla 7.3: Resultados GRASP sólo con la fase constructiva

n	$Q_{\text{máx}}$	Núm.de Instancia	g	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	113202	0.031	0.031
42	15	1	1	95403	0.031	0.015
42	15	1	2	90850	0.031	0.016
42	15	1	3	85794	0.016	0.016
42	15	1	4	81421	0.015	0.000
42	15	2	0	118221	0.031	0.031
42	15	2	1	94977	0.032	0.016
42	15	2	2	88444	0.032	0.000
42	15	2	3	84595	0.031	0.000
42	15	2	4	81794	0.047	0.015
42	15	3	0	115136	0.015	0.015
42	15	3	1	93265	0.032	0.016
42	15	3	2	88300	0.031	0.015
42	15	3	3	85183	0.031	0.000
42	15	3	4	81565	0.031	0.000

ya que en todos los casos reduce el valor de la función objetivo al compararlo con el valor que se obtiene al aplicar solamente la fase constructiva del GRASP. Se tiene, en promedio, una reducción del 3.31 % en la distancia total recorrida al aplicar la búsqueda local. Se recomienda ampliamente su utilización.

Para evaluar más el desempeño de la búsqueda local, ejecutamos el algoritmo fijando el tamaño de la cadena de vértices que puede moverse en los movimientos intrarutas e inter-rutas. Se considerará una cadena de sólo 3 vértices, los demás parámetros permanecen iguales. Se realizaron 1000 iteraciones, algunos de los resultados se observan en la tabla 7.4. Los resultados completos se encuentran en B.3 en el apéndice B.

Se puede observar que sí baja un poco el tiempo de cómputo al considerar sólo movimientos de cadenas de 3 vértices en la búsqueda local. En cuanto a la calidad de la solución en la mayoría de los casos permanece igual. Tomaremos fijo el tamaño de la cadena a mover de 3 vértices.

Se realizó un siguiente experimento para investigar si eliminando algunos tipos de vecindarios en la búsqueda local se afecta la calidad de la solución. Concretamente se suprimieron de la búsqueda local las inserciones y eliminaciones basadas en la heurística GENI de ambos tipos I y II. El tamaño de la cadena se deja libre. Algunos de los resultados se muestran en la tabla 7.5, el resto aparece en la tabla B.4 en el apéndice B. Se realizaron 1000 iteraciones.

Se observa que la búsqueda local se desempeña bien sin utilizar movimientos tipo GENI, ya que la calidad de la solución es prácticamente la misma que si se utilizan estos movimientos, pero si disminuyen los tiempos de cómputo, aunque no de forma drástica.

Evaluamos ahora el desempeño de la búsqueda local quitandole otros elementos para no hacerla tan exhaustiva y ver cómo opera. Específicamente no se considera el poder cambiar la orientación de la ruta como se hacía en los experimentos anteriores, esto es, no se considera GENI de ningún tipo y no hay chequeo del cambio de

Tabla 7.4: Resultados GRASP: Comparación fijando en 3 el tamaño de la cadena de vértices a mover en la búsqueda local

				Tamaño cadena libre			Tamaño cadena 3		
n	$Q_{\text{máx}}$	Núm.de Instancia	F	costo	Tiempo Total	Tiempo mejor sol.	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.172	0.031	111414	0.156	0.031
42	15	1	1	90934	1.125	1.078	90934	1.015	0.969
42	15	1	2	88874	1.610	0.704	88874	1.453	0.625
42	15	1	3	82973	2.906	0.266	82973	2.719	0.235
42	15	1	4	80473	4.953	0.891	80473	4.703	0.828
42	15	2	0	112218	0.360	0.078	112218	0.328	0.078
42	15	2	1	93439	2.062	1.718	93439	1.860	1.531
42	15	2	2	86529	4.625	0.703	86529	4.125	0.593
42	15	2	3	82344	9.718	3.312	82344	8.328	2.766
42	15	2	4	80011	12.719	0.172	80011	10.985	0.141
42	15	3	0	111547	0.281	0.078	111547	0.219	0.047
42	15	3	1	90763	1.375	0.031	90763	1.187	0.046
42	15	3	2	84838	2.328	0.109	84838	2.047	0.094
42	15	3	3	81690	3.703	0.844	81690	3.328	0.735
42	15	3	4	78942	4.047	2.516	78942	3.594	2.203

Tabla 7.5: Resultados GRASP sin la heurística GENI en la búsqueda local

n	$Q_{\text{máx}}$	Núm.de Instancia	F	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.172	0.032
42	15	1	1	90934	1.110	1.063
42	15	1	2	88874	1.531	0.672
42	15	1	3	82973	2.656	0.234
42	15	1	4	80473	4.844	0.813
42	15	2	0	112218	0.359	0.078
42	15	2	1	93439	1.657	1.407
42	15	2	2	86529	3.890	0.609
42	15	2	3	82344	7.735	0.297
42	15	2	4	80011	10.031	0.140
42	15	3	0	111547	0.219	0.047
42	15	3	1	90763	1.250	0.031
42	15	3	2	84838	2.141	0.110
42	15	3	3	81690	3.172	0.719
42	15	3	4	78942	3.406	1.015

Tabla 7.6: Resultados GRASP sin cambio de orientación y sin GENI

n	$Q_{\text{máx}}$	Núm.de Instancia	F	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.047	0.031
42	15	1	1	90934	0.203	0.203
42	15	1	2	88874	0.297	0.141
42	15	1	3	82973	0.453	0.062
42	15	1	4	80605	0.734	0.031
42	15	2	0	112218	0.094	0.032
42	15	2	1	93439	0.328	0.281
42	15	2	2	86529	0.703	0.125
42	15	2	3	82344	1.203	0.063
42	15	2	4	80011	1.454	0.032
42	15	3	0	111547	0.062	0.031
42	15	3	1	90763	0.234	0.000
42	15	3	2	84838	0.407	0.188
42	15	3	3	81690	0.562	0.140
42	15	3	4	79074	0.531	0.172

orientación. Realizando 1000 iteraciones del GRASP algunos de los resultados se pueden observar en la tabla 7.6 y el resto en la tabla B.5 en el apéndice B.

Se observa que aquí sí se logra una reducción de tiempo muy significativa, los movimientos tipo GENI y los cambios de orientación consumen mucho tiempo en los cálculos. La calidad de la solución prácticamente no se ve afectada por estos cambios, pero el tiempo sí. Por lo que se recomienda no utilizar movimientos tipo GENI ni checar cambios de orientación en la búsqueda local.

Se realizaron experimentos computacionales para probar la efectividad de la búsqueda local super rápida, la cual en cada subvecindario definido en la búsqueda

Tabla 7.7: Resultados GRASP con búsqueda local super rápida

n	$Q_{\text{máx}}$	Núm.de Instancia	F	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.156	0.031
42	15	1	1	90934	1.031	1.000
42	15	1	2	88874	1.500	0.672
42	15	1	3	82973	2.688	0.266
42	15	1	4	80473	4.468	0.843
42	15	2	0	112218	0.313	0.079
42	15	2	1	93439	1.937	1.609
42	15	2	2	86529	4.266	0.657
42	15	2	3	82344	9.094	3.016
42	15	2	4	80011	11.266	0.157
42	15	3	0	111547	0.219	0.047
42	15	3	1	90763	1.234	0.047
42	15	3	2	84838	2.156	0.094
42	15	3	3	81690	3.360	0.797
42	15	3	4	78942	3.594	2.250

local rápida guarda no sólo si está activo o no, también el costo y los elementos del mejor movimiento que se puede hacer en este subvecindario, para realizar la búsqueda local más rápidamente. El algoritmo se ejecuta sin límite en el tamaño de la cadena que se puede mover, utilizando GENI de dos tipos y sí se considera cambio de orientación. Análogamente a los experimentos anteriores, se realizan 1000 iteraciones del GRASP. Algunos resultados se muestran en la tabla 7.7, los resultados completos se presentan en la tabla B.6 y se deben comparar con los reportados en B.3 del apéndice B.

Se observa que la calidad de la solución es la misma, pero sí se reduce el tiempo

de cómputo necesario, se recomienda su utilización.

Teniendo en cuenta los resultados obtenidos, no se recomienda considerar una búsqueda local tan amplia como la que inicialmente se había propuesto, concretamente vemos que los movimientos que generalizan el GENI consumen mucho tiempo de computación y la reducción obtenida en costo, esto es, la mejoría en la calidad de la solución obtenida, no es demasiado significativa. Se propone considerar solamente los movimientos más comunes, estos son, cambio de una cadena de vértices dentro de una misma ruta, intercambio de cadenas de vértices en rutas diferentes y eliminación de una cadena de vértices para ser insertada en otra ruta en su versión estándar, esto es, con eliminaciones e inserciones clásicas.

Calibración de parámetros α y β

Mostramos enseguida el efecto de los parámetros α y β en la ejecución del GRASP. Recordemos que α es el valor para la generación de la lista restringida de candidatos en el GRASP y β es el valor que controla la diversidad de soluciones. Se ejecutó el GRASP considerando diferentes valores de estos parámetros. Se realizaron 1000 iteraciones sin considerar límite al tamaño de la cadena a mover en la búsqueda local, sin utilizar movimientos tipo GENI en la búsqueda local y sin considerar cambio de orientación, esto dado los resultados reportados anteriormente.

En los siguientes resultados el punto de partida para la generación de los puntos semilla en el algoritmo de Fisher y Jaikumar es fijo, o sea, siempre se generan los mismos puntos semilla en una instancia dada. En el caso de $\alpha = \beta = 0$ sólo se genera una solución. Sí varían estos parámetros si se generan varias soluciones iniciales en el GRASP. Se presentan algunos de los resultados de la ejecución en la tabla 7.8, los resultados completos se muestran en las tablas B.7 y B.8 del apéndice B.

Las soluciones mejoran considerablemente cuando se penalizan las frecuencias (factor β) y se consideran varios elementos en la lista restringida de candidatos en la construcción de la solución inicial (factor α). Cuando $\alpha = 0$ solo se genera la

Tabla 7.8: Resultados GRASP con diferentes valores para α y β

				$\alpha = 0, \quad \beta = 0$			$\alpha = 0.5, \quad \beta = 0$			$\alpha = 0, \quad \beta = 1$			$\alpha = 0.5, \quad \beta = 1$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo		Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.	costo	Total	mejor s.	costo	Total	mejor s.
42	15	1	0	111414	0.000	0.000	111256	0.375	0.047	111256	0.047	0.015	111256	0.672	0.031
42	15	1	1	97749	0.000	0.000	93220	1.093	0.140	91380	0.125	0.094	90774	2.063	1.063
42	15	1	2	92565	0.015	0.015	88874	1.469	0.125	88874	2.360	0.078	88874	2.250	0.890
42	15	1	3	87247	0.015	0.015	82996	1.735	1.000	82407	3.125	0.171	82533	3.156	2.578
42	15	1	4	83027	0.031	0.031	83027	3.109	0.047	80360	4.235	1.047	80363	4.344	1.203
42	15	2	0	113875	0.000	0.000	112218	0.563	0.016	113136	0.031	0.015	112218	0.172	0.047
42	15	2	1	93997	0.015	0.015	93439	1.297	0.829	90900	2.172	0.453	90900	2.250	0.672
42	15	2	2	87675	0.031	0.031	87675	1.985	0.016	86077	3.312	0.984	86077	3.360	0.032
42	15	2	3	84092	0.031	0.031	84092	2.937	0.031	82344	4.547	0.125	82344	4.391	1.141
42	15	2	4	82041	0.031	0.031	82041	3.172	0.031	80011	5.484	1.797	80011	5.594	2.922
42	15	3	0	111547	0.000	0.000	111547	0.281	0.016	111547	0.032	0.016	111547	0.328	0.015
42	15	3	1	92306	0.015	0.015	90763	0.907	0.063	90763	1.843	0.328	90763	1.844	0.547
42	15	3	2	89393	0.016	0.016	85707	1.390	0.250	84838	2.469	2.282	84838	2.438	1.750
42	15	3	3	85838	0.015	0.015	81365	2.266	0.157	80940	3.312	1.016	80778	3.375	3.313
42	15	3	4	79120	0.032	0.032	79120	2.703	0.031	78637	4.406	0.453	78683	4.391	0.484

lista restringida de candidatos con un solo elemento, pero al variar este parámetro a $\alpha = 0.5$ hay mejores resultados. De igual forma al penalizar la frecuencia con la que un vértice se asigna a una ruta se obtiene más variedad de soluciones y permite obtener mejores aproximaciones a la mejor solución. También se observa, como era de esperarse, que el tiempo de cómputo aumenta al considerar estos parámetros diferentes de cero, pero la reducción de costos vale la pena el esfuerzo computacional extra. Se observa que $\alpha = 0.5$ y $\beta = 1$ proporciona buenos resultados.

En todos los experimentos reportados hasta el momento, no se utilizó el encadenamiento de trayectorias, presentamos enseguida cómo varía la calidad de las soluciones obtenidas si aplicamos este post-procesamiento.

Experimentos para evaluar los procedimientos diseñados en el encadenamiento de trayectorias (fase II del HGET)

Los siguientes experimentos que se realizaron están enfocados a evaluar el desempeño del encadenamiento de trayectorias, es decir, la segunda fase del algoritmo HGET propuesto.

En estos experimentos se utilizaron las instancias del tercer tipo.

Se realizó un primer experimento para investigar el efecto de la cantidad de soluciones de buena calidad que se utilizan en el encadenamiento de trayectorias, esto es, del tamaño del conjunto $Setofbest$. Se probó con diferentes tamaños de este conjunto de soluciones y los resultados se presentan en la tabla 7.9. Se consideraron los valores $\alpha = 0.5$ y $\beta = 1$ y se realizaron 1000 iteraciones del GRASP.

Se observa que en todos los casos se obtiene un beneficio al aplicar el encadenamiento de trayectorias respecto a los resultados que se obtienen sin aplicar este postprocesamiento. Este beneficio es mas significativo en las instancias de mayor tamaño en donde la reducción es más grande en términos relativos. También se observa que al aumentar el número de buenas soluciones que se utilizan en la Fase II se mejora la calidad de la solución, pero con el número "mediano"de 15 buenas

Tabla 7.9: Resultados HGET para diferentes tamaños del conjunto *SetofBest*

			HGET Número de soluciones para el encadenamiento de trayectorias			
n	Inst.	GRASP	10	15	20	40
42	1	6425.17	6425.17	6425.17	6425.17	6425.17
	2	6173.31	6173.31	6173.31	6173.31	6173.31
	3	6355.62	6355.62	6355.62	6355.62	6355.62
	4	6902.71	6902.71	6902.71	6902.71	6902.71
	5	6832.63	6832.63	6832.63	6832.63	6832.63
62	1	9317.89	9297.57	9297.57	9296.55	9296.55
	2	8859.51	8799.13	8799.13	8780.14	8780.14
	3	8813.65	8813.65	8813.65	8813.65	8813.65
	4	9089.24	9089.24	9089.24	9089.24	9089.24
	5	8862.58	8838.03	8838.03	8838.03	8838.03
82	1	10941.33	10788.73	10788.73	10788.73	10785.72
	2	10438.98	10435.70	10430.65	10430.65	10430.13
	3	10157.40	10087.66	10086.86	10086.86	10086.86
	4	10602.39	10550.06	10550.06	10550.06	10550.06
	5	10740.17	10668.74	10631.00	10631.00	10617.68
102	1	13365.72	13249.25	13168.01	13168.01	13160.83
	2	13984.19	13839.54	13827.75	13827.75	13809.51
	3	11992.43	11898.51	11898.51	11898.51	11898.51
	4	12556.61	12426.21	12398.57	12314.94	12314.94
	5	12690.20	12599.44	12569.09	12569.09	12537.30

soluciones se obtienen buenos resultados y se consume menos tiempo de cómputo.

Como conclusión se puede mencionar que el post–procesamiento diseñado basado en el encadenamiento de trayectorias es de gran beneficio en este caso, ya que logra mejorar considerablemente el valor de la función objetivo en todos los experimentos realizados.

7.2.2 Experimentos para evaluar el algoritmo BEV-RF

La segunda metodología propuesta para el problema mono-objetivo fue un algoritmo basado en la metaheurística de búsqueda por entornos variables detallada en el capítulo 4.

Presentamos ahora los experimentos realizados para evaluar el desempeño de esta metodología. Al igual que con el método híbrido primero se resolvieron instancias chicas para comprobar que se alcanzaba el óptimo. Posteriormente se utilizaron las instancias del tercer tipo para calibrar parámetros.

Validación del algoritmo. Comparación con Cplex

Primeramente se comprobó que el algoritmo es capaz de obtener el óptimo al comparar las soluciones obtenidas en instancias pequeñas resueltas con software comercial. Se probó en las instancias del primer tipo y los resultados obtenidos se presentan en la tabla 7.10 para 10, 12 y 15 nodos. En todos los casos se logra llegar al mismo valor de la función objetivo, por lo que se reporta sólo el tiempo de cómputo utilizado en segundos.

En estas pruebas se toma como $r = 10$ el tamaño máximo de vecindad en el proceso de agitación y se consideraron 1000 iteraciones del BEV-RF.

La búsqueda local que se utiliza es la misma que en algoritmo HGET, así que aquí ya no se realiza una evaluación del desempeño de ésta, solamente se ajusta el

Tabla 7.10: Tiempos BEV-RF en instancias aleatorias $n = 10, 12, 15$

	$n = 10$		$n = 12$		$n = 15$	
g	CPLEX	BEV-RF	CPLEX	BEV-RF	CPLEX	BEV-RF
1	3.42	2.8	11.11	7.25	284	10.18
2	8.99	3.1	160	9.55	53278	12.77
3	54.95	3.5	1658	10.87	110621	16.49
4	79.49	3.8	10295	15.37	615973	35.52

parámetro r, lo cual se verá en el experimento que se explica a continuación.

Calibración de parámetros del BEV-RF

En el siguiente experimento se utilizaron instancias del tercer tipo. Se consideraron diferentes valores del tamaño máximo que puede tener el vecindario en el proceso de agitación, el parámetro r. Algunos de los resultados de BEV-RF para diferentes valores de r se muestran en la tabla 7.11, una tabla más completa se muestra en B.9 en el apéndice B . En todos los casos se realizaron 1000 iteraciones de BEV-RF.

Se observa que conforme el tamaño de r aumenta, se obtienen generalmente mejores resultados, pero con tiempos de cómputo mayores; $r = 13$ es una elección que reporta buenos resultados con tiempos considerables.

Presentamos enseguida comparaciones entre estos algoritmos y otros reportados en la literatura para verificar la calidad de la soluciones obtenidas.

7.2.3 Comparaciones con otros algoritmos

Como hemos dicho anteriormente, en la literatura científica no encontramos métodos desarrollados específicamente para el tipo de problema abordado en este trabajo. Por otra parte, el problema estudiado pudiera ser modelado como un caso particular del PVRP. Entonces, con el propósito de evaluar el desempeño de los

Tabla 7.11: Resultados BEV-RF para diferentes valores de r

n	Inst.	$r = 5$	$r = 13$	$r = 20$
42	1	6425.17	6425.17	6425.17
	2	6173.31	6173.31	6173.31
	3	6355.62	6355.62	6355.62
	4	6902.71	6902.71	6902.71
	5	6832.63	6832.63	6832.63
62	1	9338.92	9316.82	9298.06
	2	8851.1	8766.68	8765.2
	3	8865.25	8813.65	8893.57
	4	9089.24	9089.24	9089.24
	5	8946.49	8942.57	8942.57
82	1	10931.77	10880.42	10788.73
	2	10520.13	10430.65	10430.65
	3	10182.39	10075.53	10075.53
	4	10555.07	10550.06	10550.06
	5	10769.95	10662.9	10662.21
102	1	13302.32	13279.87	13111.88
	2	13884.61	13799.18	13799.18
	3	11982.54	11902.96	11913.58
	4	12373.72	12373.72	12303.52
	5	12675.54	12575.35	12580.2

algoritmos que proponemos, realizamos las modificaciones necesarias para modelar el problema como un PVRP y así poder realizar comparaciones computacionales con algoritmos desarrollados para el PVRP.

Uno de los mejores métodos para problemas de ruteo periódicos está basado en Búsqueda Tabú y fue desarrollado por J. Cordeau et al. [22], (CGL). Otro algoritmo con el que comparamos está basado en una Búsqueda Dispersa desarrollado por Alegre et al. [7], (ALP) y el último método para comparar está basado en una búsqueda por entornos variables desarrollado por V. Hemmelmayr et al. [51], (HDH). Nuestras instancias fueron enviadas a cada uno de estos autores, los cuales las resolvieron utilizando sus respectivos algoritmos. Agradecemos a ellos su tiempo y disposición.

El algoritmo CGL fue ejecutado en una Dell Precision T7400 con cuatro procesadores a 3.0 GHz CPU, sólo se utilizó un procesador en las pruebas. El algoritmo ALP fue ejecutado en una Corel 8400 PC a 2.66 GHz CPU; el algoritmo HDH se ejecutó en una 64 bits dual core PC, con 3.2 GHz CPU. En la tabla 7.12 se encuentra el resumen de los resultados obtenidos para cada método, específicamente la distancia total (Dist) y los tiempos de computación en segundos. En el método híbrido HGET se utilizó $\alpha = 0.5$ y $\beta = 1$ conservando las 12 mejores soluciones del GRASP para el encadenamiento de trayectorias. En BEV-RF se utiliza $r = 13$ y en los dos algoritmos se realizaron 1000 iteraciones.

De la tabla 7.12 se puede observar que los cinco algoritmos obtienen resultados similares para el caso de instancias pequeñas (n = 42 y 62). En las instancias más grandes (n = 82 y 102) los resultados obtenidos con el HGET y el BEV-RF son competitivos. Sólo en dos de las 20 instancias alguno de los dos algoritmos es superado. CGL obtiene los mejores resultados en 8 instancias, ALP en 8 de ellas, HDH en 5, HGET en 16 y el BEV-RF en 11. Los algoritmos HGET, ALP y HDH utilizan tiempos de cómputo similares, en tanto que CGL necesita menos tiempo y el BEV-RF más tiempo para alcanzar la solución, sobre todo en las instancias más grandes.

Tabla 7.12: Resultados obtenidos con diferentes algoritmos y HGET y BEV-RF

n	Inst.	CGL		ALP		HDH		HGET		BEV-RF	
		Dist	Tiempo	Dist	Tiempo	Dist	Tiempo	Dist	Tiempo	Dist	Tiempo
42	1	6425,17	36,00	6425,17	32,9	6425,17	79,60	6425.17	29,2	6425.17	145.57
	2	6173,31	36,60	6314,66	30,9	6176,67	82,90	6173.31	28,8	6173.31	138.18
	3	6355,62	40,90	6355,62	32,5	6355,62	81,80	6355.62	29,6	6355.62	137.81
	4	6902,71	34,80	6902,71	34,8	6921,95	84,20	6902.71	26,8	6902.71	116.01
	5	6832,63	29,40	6832,63	30,5	6832,63	86,60	6832.63	23,9	6832.63	117.18
62	1	9298,06	57,60	9317,89	98,0	9335,90	166,10	9296.55	81,4	9316.82	228.97
	2	8765,20	61,80	8823,87	91,4	8785,07	171,30	8765.20	81,1	8766.68	274.99
	3	8833,46	58,20	8885,98	94,5	8849,44	177,00	8813.65	70,8	8813.65	274.99
	4	9089,24	40,90	9089,24	77,2	9089,24	173,80	9089.24	71,8	9089.24	231.64
	5	8838,03	68,40	8851,54	116,7	8878,02	162,60	8838.03	95,7	8942.57	633.22
82	1	10779,09	89,40	10912,78	206,0	10826,58	276,50	10788.73	162,9	10880.42	416.48
	2	10459,80	93,00	10480,6	217,0	10501,43	283,20	10430.65	203,5	10430.65	395.53
	3	10147,79	98,40	10075,53	209,7	10138,96	287,00	10076.22	199,2	10075.53	543.66
	4	10555,07	72,00	10550,06	233,5	10619,69	289,00	10550.06	176,9	10550.06	641.90
	5	10611,28	91,20	10592,93	217,1	10618,36	307,00	10663.80	209,5	10662.9	411.15
102	1	13312,92	111,60	13285,43	376,5	13191,64	434,20	13203.90	349,1	13279.87	811.51
	2	13887,50	91,80	13834,5	388,5	13893,59	464,90	13827.75	325,2	13799.18	598.58
	3	11920,98	117,00	11957,09	467,8	11981,48	455,30	11898.51	389,4	11902.96	595.28
	4	12515,18	126,00	12423,94	454,1	12482,99	438,00	12354.58	402,1	12373.72	595.28
	5	12591,00	127,80	12573,39	519,2	12640,49	439,20	12518.62	416,1	12575.35	648.44

En la tabla 7.13 se muestra el porcentaje (%) de desviación de los resultados obtenidos con la mejor solución conocida para cada una de las instancias y la media de la desviación. Las mejores soluciones se obtuvieron ejecutando ALP, HDH y HGET con más tiempo computacional. Se puede observar que el método HGET logra llegar a la mejor solución en 13 de las 20 instancias, en tanto que el BEV-RF logra llegar a la mejor solución en 10 instancias. La desviación media en el caso del método HGET es menor al 0.17 %, es la menor de todos algoritmos; le sigue el BEV-RF con una desviación media menor al 0.35 %.

7.2.4 Análisis del efecto de la fexibilidad de las fechas de entrega: comparación con la solución actual de la compañía

El siguiente experimento se realizó para analizar el efecto o la influencia de la flexibilidad en las fechas de entrega en la distancia total recorrida, esto es, comparar con la forma en que la compañía actualmente realiza su ruteo. Para ello se comparan los resultados que se obtienen con nuestros algoritmos considerando g=0, esto es, entregas día a día, y g=1, el adelanto máximo de un día. Recordemos que el ruteo día a día es como la empresa actualmente realiza sus entregas, por eso nos interesa esta comparación. Los experimentos se realizaron en las instancias del tercer tipo y los resultados se muestran en la tabla 7.14 con el porcentaje de reducción obtenido en la distancia total recorrida (%).

Se observa que permitir cierta flexibilidad en las fechas de entrega, en este caso permitir máximo un día de adelanto, da lugar a una reducción importante en la distancia total recorrida de alrededor del 20 %. Esto impacta directamente la economía de la compañía, por lo que resulta recomendable permitir este adelanto. Además, cabe señalar que el modelo con $g = 0$ (ruteo día a día), que es como la compañía actualmente opera, es equivalente a cinco CVRP independientes, uno por cada día de la semana de alrededor de 20 órdenes cada uno para las instancias mas

Tabla 7.13: Resultados del % de desviación a la mejor solución encontrada

n	Inst.	CGL	ALP	HDH	HGET	BEV-RF	Mejor sol.
		% de desviación a la mejor solución conocida					
42	1	0	0	0	0	0	6425.17
	2	0	2.29	0.05	0	0	6173.31
	3	0	0	0	0	0	6355.62
	4	0	0	0.28	0	0	6902.71
	5	0	0	0	0	0	6832.63
62	1	0.02	0.23	0.42	0	0.21	9296.55
	2	0	0.67	0.23	0	0.01	8765.20
	3	0.22	0.82	0.41	0	0	8813.65
	4	0	0	0	0	0	9089.24
	5	0	0.15	0.45	0	1.16	8838.03
82	1	0.57	1.82	1.02	0.65	1.49	10717.63
	2	0.39	0.59	0.79	0.1	0.10	10419.42
	3	0.72	0	0.63	0.00	0	10075.53
	4	0.05	0	0.66	0	0	10550.06
	5	0.48	0.31	0.64	0.96	0.95	10560.64
102	1	1.77	1.56	0.84	0.92	1.49	13081.66
	2	0.63	0.25	0.67	0.18	0	13799.18
	3	0.19	0.49	0.70	0	0.30	11898.51
	4	1.72	0.98	1.46	0.41	0.56	12303.52
	5	0.86	0.72	1.41	0.27	0.72	12484.05
media		0.381	0.543	0.523	0.174	0.349	

Tabla 7.14: Comparaciones de la distancia recorrida con y sin flexibilidad en la fecha de entrega

n	Inst.	Distancia total		
		$g = 0$	$g = 1$	% reducción
42	1	8128.46	6425.17	20.95
	2	7935.32	6173.31	22.20
	3	8190.05	6355.62	22.40
	4	8784.39	6902.71	21.42
	5	8354.68	6832.63	18.22
62	1	11590.32	9296.55	19.79
	2	10537.84	8765.20	16.82
	3	10854.58	8813.65	18.80
	4	11214.50	9089.24	18.95
	5	11323.35	8838.03	21.95
82	1	12980.12	10788.73	16.88
	2	12683.18	10430.65	17.76
	3	12725.47	10076.22	20.81
	4	13561.32	10550.06	22.20
	5	13356.78	10663.80	20.16
102	1	16015.65	13203.90	17.55
	2	16996.61	13827.75	18.64
	3	14838.80	11898.51	19.81
	4	15097.29	12354.58	18.16
	5	15165.98	12518.62	17.45

grandes, en este caso el algoritmo estará cerca del óptimo en cada uno de estos subproblemas, ya que son instancias con pocos nodos. Como la complejidad del problema es mayor cuando $g = 1$ que cuando $g = 0$, es posible obtener aún mejores soluciones con costos más bajos que representen un mayor beneficio para la compañía, de donde se recomienda ampliamente permitir este adelanto de fechas de entrega.

7.2.5 Resumen

En general, podemos comentar que los dos algoritmos propuestos para el problema con un solo objetivo trabajan bien, logran llegar al óptimo en instancias chicas y son competitivos en instancias más grandes.

De la evaluación de la búsqueda local concluimos que sin necesidad de hacerla tan exhaustiva se pueden obtener buenos resultados en tiempos de cómputo razonables. Otra observación importante es que se logra una reducción considerable en la distancia total recorrida cuando se permite adelantar los pedidos un día; es conveniente para la compañía permitir esta flexibilidad en las fechas de entrega.

7.3 Resultados computacionales para el modelo biobjetivo

Presentamos en esta sección los resultados de los experimentos computacionales que se realizaron con las metodologías desarrolladas para abordar el problema con los dos objetivos simultáneamente, esto es, minimizar la distancia total recorrida y los costos de almacenamiento. Para resolver el problema desde este enfoque se propusieron dos algoritmos heurísticos de solución en el capítulo 6, uno que acopla la metaheurística MOAMP basado en una búsqueda por entornos variables para el problema biobjetivo (MO-BEV) y otro basado en una implementación adecuada del algoritmo evolutivo NSGA-II, que llamamos NSGA-RF. Presentamos enseguida algunos resultados de experimentos computacionales realizados para comprobar la

efectividad de estos algoritmos.

Primeramente se probaron los dos algoritmos en las instancias aleatorias de 10 y 12 nodos que se resolvieron con ε–restricción en el capítulo 5, se obtuvieron los mismos frentes de Pareto que con este método exacto.

Todos los resultados que se presentan enseguida se obtuvieron al aplicarse los algoritmos a instancias del tercer tipo.

En la sección 7.3.1 se presentan los resultados al evaluar el desempeño de cada una de las fases del algoritmo MO-BEV, recordemos que este algoritmo consta de tres fases, deseamos saber cómo influye cada una de ellas en los resultados obtenidos. En la sección 7.3.2 se presenta la comparación entre los dos algoritmos MO-BEV y NSGA-RF, comparando número de soluciones no dominadas, aproximaciones a los extremos de la frontera de Pareto y se muestran algunos resulados gráficamente con los frentes obtenidos por ambas metodologías.

7.3.1 Experimentos para evaluar el desempeño de las fases del algoritmo MO-BEV

En esta sección presentamos los resultados de experimentos realizados para evaluar cada una de las fases del MO-BEV. Como se comentó en el capítulo 6, este método es un algoritmo de tres fases. En la primera se optimiza cada uno de los objetivos por separado, en la segunda se optimiza una función $F_\lambda(S)$ y en la tercera fase se exploran los vecindarios de las soluciones no dominadas obtenidas en las dos fases anteriores. Los primeros experimentos se desarrollaron para evaluar el desempeño de cada una de estas fases. Para ello se ejecuta el algoritmo en un conjunto de instancias y se compara el número de soluciones no dominadas obtenidas en cada una de las fases del algoritmo.

Como se recordará, primeramente se requiere construir una buena solución inicial considerando solamente el objetivo de la distancia total recorrida. En nues-

tra implementación esta solución se construye utilizando el algoritmo híbrido de GRASP con encadenamiento de trayectorias (HGET), descrito en 4.1, considerando 1000 iteraciones del GRASP y las 15 mejores soluciones para el encadenamiento de trayectorias. De igual forma, para construir la solución inicial tomando en cuenta sólo el objetivo del almacenamiento, se considera que las órdenes se entregan el día inicialmente solicitado por los distruibuidores, para tener un almacenamiento cero, y se obtiene una buena solución en cuanto a distancia utilizando el método HGET con los mismos parámetros de 1000 iteraciones para el GRASP y las 15 mejores soluciones para el encadenamiento de trayectorias.

Con las soluciones iniciales construidas se optimizan cada uno de los objetivos por separado utilizando el algoritmo BEV-RFB considerando un número máximo de iteraciones sin mejora de 250 y en la fase de Agitación el parámetro $r_{\text{máx}}$, que es el número máximo permitido para cortes en las grandes cadenas, se toma igual a 10.

En la segunda fase se optimiza la función $F_\lambda(S)$ con el algoritmo BEV-RFB, aquí intervienen también el número máximo de iteraciones sin mejora y el parámetro $r_{\text{máx}}$, en este experimento se consideraron como 20 y 5, respectivamente.

Por último en la tercera fase se explora el vecindario de cada solución no dominada buscando nuevas soluciones no dominadas para completar el frente de Pareto, esta exploración se realiza utilizando los cuatro movimientos descritos en 6.2.

En la tabla 7.15 se muestran el número de soluciones no dominadas encontradas en cada fase así como el tiempo de ejecución para las instancias más pequeñas. La tabla completa se presenta en C.1 en el apéndice C.

Se observa que en la primera fase se obtienen pocos puntos, pero cabe resaltar que se logra aproximar muy bien desde esta primera fase los extremos del intervalo del conjunto de puntos no dominados, esto es, los mejores valores de los objetivos por separado se aproximan bien desde el principio. Sin embargo no se logran buenas aproximaciones en el interior de la curva de eficiencia en esta primera fase. En

Tabla 7.15: Número de puntos no dominados obtenidos al finalizar cada fase del MO-BEV

		Número de puntos no dominados en cada fase y tiempo					
	Núm.	Primera fase		Segunda fase		Tercera fase	
n	Inst.	No.puntos ND	tiempo	No.puntos ND	tiempo	No.puntos ND	tiempo
42	1	9	134	46	105	46	0.063
	2	14	137	33	133	35	0.047
	3	10	138	25	90	27	0.032
	4	12	130	62	114	67	0.094
	5	9	108	32	97	46	0.062

la segunda fase se mejora notablemente la aproximación a la curva de eficiencia en el interior, se logra diversificación obteniendo un conjunto denso de puntos de buena calidad y se aumenta considerablemente el número de puntos no dominados encontrados. La intensificación se logra en la tercera fase donde se obtienen más y mejores puntos no dominados. Con excepción de la primera instancia, en todas las demás instancias se logra aumentar el conjunto de puntos no dominados. Se observa que en las instancias más grandes se aumenta considerablemente el número de puntos encontrados en esta fase respecto a los obtenidos hasta la fase dos.

En cuanto a los tiempos de ejecución, las fases I y II necesitan tiempos similares de ejecución y el tiempo aumenta conforme aumenta el tamaño de las instancias. La tercera fase, en cambio, es mucho más rápida y en pocos segundos logra finalizar la exploración de los puntos no dominados obteniendo buenos resultados. Recordemos que estamos utilizando lo que denominamos búsqueda local rápida, lo cual acelera la búsqueda en esta tercera fase, esto se refleja en los resultados obtenidos.

Se muestran también de forma gráfica algunos de estos resultados para la instancia $\texttt{Instancia}82_2$ en la gráfica 7.2. Para las instancias $\texttt{Instancia}82_5$, $\texttt{Instancia}102_1$ las gráficas correspondientes se encuentran en C.1 en el apéndice C.

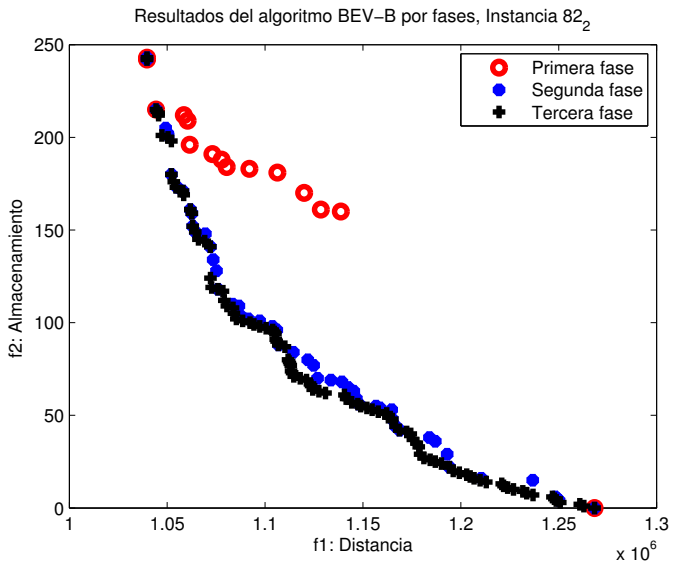


Figura 7.2: Resultados por fases del algoritmo MO-BEV para la instancia Instancia82_2

7.3.2 Comparación de resultados obtenidos con MO-BEV y NSGA-RF

Los experimentos que se describen en esta sección se realizaron para comparar los resultados obtenidos con las dos metodologías propuestas, los algoritmos MO-BEV y NSGA-RF .

Se consideraron diferentes parámetros para MO-BEV para evaluar su comportamiento en el conjunto de soluciones no dominadas que se obtiene. En particular se consideró el máximo de iteraciones sin mejora como 20, 50 y 100 y $r_{\text{máx}}$ como 5 y 10. Para tener una comparación justa entre ambos algoritmos consideramos el criterio de paro para NSGA-RF como el tiempo de ejecución de MO-BEV. Esto es, para una instancia dada primero se ejecuta MO-BEV, tomamos el tiempo de ejecución y enseguida se ejecuta el NSGA-RF para la misma instancia durante el mismo tiempo. De esta forma los dos algoritmos son ejecutados por el mismo período de tiempo. El tamaño de la población para el algoritmo NSGA-RF se considera como el doble del número de puntos no dominados obtenidos por el algoritmo MO-BEV

Tabla 7.16: Número de puntos no dominados obtenidos con MO-BEV y con NSGA-RF

		Número de puntos no dominados					
	Núm.	(20,5)		(50,10)		(100,10)	
n	Inst.	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF
42	1	46	47	49	47	49	47
	2	35	36	36	34	36	34
	3	27	23	28	31	28	31
	4	67	56	65	68	68	58
	5	46	46	50	50	51	48

y la probabilidad de mutación se considera igual a 0.1, se eligió este valor después de realizar algunos experimentos computacionales variando esta probabilidad que se presentan en la tabla C.7.

Enseguida presentamos algunos de los resultados obtenidos. Primero se presentan el número de puntos no dominados encontrados por ambos algoritmos variando algunos parámetros, la combinación (20,5) corresponde a un máximo de 20 iteraciones sin mejora y $r_{\text{máx}} = 5$ en la fase II del MO-BEV, de igual forma se presentan resultados para (50,10) y (100,10). En la tabla 7.16 se muestran algunos de estos resultados y la tabla completa se encuentra en C.2 en el apéndice C.

Se observa que en algunos casos NSGA-RF logra obtener un mayor número de soluciones no dominadas que MO-BEV y en otros es a la inversa, en general no podemos establecer que un algoritmo obtiene mas puntos que otro. También se observa que en general, al aumentar el número máximo permitido de iteraciones sin mejora y el número máximo de cortes permitido en BEV-RFB en la fase II de este último, no se aumenta considerablemente el número de soluciones no dominadas encontradas. Cabe mencionar que al aumentar estos parámetros aumenta el tiempo utilizado por el algoritmo, como se verá más adelante, por lo que resulta interesante

observar los resultados de NSGA-RF, el cual, en la mayoría de los casos, sí aumenta el número de puntos no dominados encontrados, sobre todo en instancias grandes.

Más adelante se presentan algunos resultados gráficamente de las aproximaciones a la curva de eficiencia por los dos algoritmos y donde se observa que los puntos obtenidos por MO-BEV logran una mayor diversificación del frente de Pareto. Esto es, NSGA-RF obtiene, en general, puntos no dominados que no se distribuyen en toda la curva de eficiencia, hay regiones donde no se explora a fondo el frente de Pareto, estos puntos son de buena calidad en el sentido de la intensificación, aproximan bien una región de la frontera de Pareto. En cambio, la aproximación obtenida por MO-BEV, en general está más distribuida sobre la curva de eficiencia con mayor diversificacin y la intensificación es buena.

Se presentan a continuación algunos de los resultados respecto a los puntos extremos del intervalo del conjunto de puntos no dominados alcanzado por ambos algoritmos. El extremo derecho corresponde al punto con mejor valor de la función del almacenamiento, en tanto que el extremo izquierdo corresponde al punto con mejor valor de la función distancia, ambos en el frente de Pareto obtenido.

Los resultados para las instancias más pequeñas se muestran en 7.17 y la tabla completa se encuentra en C.3 en el Apéndice C.

De igual manera, se muestran algunos de los resultados respecto al otro punto extremo del intervalo del conjunto de puntos no dominados, al igual que los resultados anteriores, se presenta aqui sólo una muestra en la tabla 7.18 y la tabla completa se encuentra en C.4 en el Apéndice C.

Se puede observar que el punto con mejor valor en cuanto al almacenamiento, el extremo derecho, no cambia para los diferentes parámetros del MO-BEV, permanece igual; pero los puntos encontrados por NSGA-RF sí tienen cambios en las instancias más grandes, en la mayoría de los casos, mejora este punto extremo al ejecutar el algoritmo por más tiempo. En las instancias de 42 órdenes los puntos coinciden por ambos algoritmos, pero en todas las demás instancias el algoritmo MO-BEV obtiene

Tabla 7.17: Extremo derecho del frente de Pareto (puntos con mejor valor de la función de almacenamiento) obtenidos con MO-BEV y con NSGA-RF

		Extremo derecho del frente de Pareto					
	Núm.	(20,5)		(50,10)		(100,10)	
n	Inst.	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF
42	1	(812824 , 0)	(812824 , 0)	(812824 , 0)	(812824 , 0)	(812824 , 0)	(812824 , 0)
	2	(793506 , 0)	(793506 , 0)	(793506 , 0)	(793506 , 0)	(793506 , 0)	(793506 , 0)
	3	(818979 , 0)	(818979 , 0)	(818979 , 0)	(818979 , 0)	(818979 , 0)	(818979 , 0)
	4	(878414 , 0)	(878414 , 0)	(878414 , 0)	(878414 , 0)	(878414 , 0)	(878414 , 0)
	5	(835442 , 0)	(835442 , 0)	(835442 , 0)	(835442 , 0)	(835442 , 0)	(835442 , 0)

Tabla 7.18: Extremo izquierdo del frente de Pareto (puntos con mejor valor de la función distancia) obtenidos con MO-BEV y con NSGA-RF

		Extremo izquierdo del frente de Pareto					
	Núm.	(20,5)		(50,10)		(100,10)	
n	Inst.	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF
42	1	(642495 , 94)	(642495 , 94)	(642495 , 94)	(643006 , 74)	(642495 , 94)	(643006 , 74)
	2	(617309 , 115)	(617344 , 116)	(617309 , 115)	(617309 , 115)	(617309 , 115)	(617309 , 115)
	3	(635539 , 75)	(666938 , 54)	(635539 , 75)	(635539 , 75)	(635539 , 75)	(635539 , 75)
	4	(690249 , 142)	(697973 , 139)	(690249 , 142)	(690249 , 142)	(690249 , 142)	(698269 , 107)
	5	(683240 , 119)	(698153 , 78)	(683240 , 119)	(691482 , 99)	(683240 , 119)	(683240 , 119)

Tabla 7.19: Tiempo utilizado por MO-BEV y NSGA-RF con los diferentes parámetros

		Tiempo		
n	Núm.Inst.	(20,5)	(50,10)	(100,10)
42	1	239	794	1590
	2	270	667	645
	3	228	492	672
	4	244	782	695
	5	205	607	1089

mejores aproximaciones a este punto extremo que NSGA-RF.

La situación en el mejor punto en cuanto la distancia, el extremo izquierdo, es similar; aquí el algoritmo MO-BEV supera en todos los casos los puntos encontrados por NSGA-RF. Hay pequeñas variaciones en algunos de los puntos encontrados por MO-BEV al cambiar los parámetros, pero no son muy significativas. También hay variaciones en los puntos encontrados por NSGA-RF al aumentar el tiempo de ejecución, pero no logra mejorar las aproximaciones de MO-BEV. Se logra explorar más a fondo esta región del frente de Pareto con el algoritmo MO-BEV al encontrar más soluciones no dominadas cerca del extremo izquierdo.

En la tabla 7.19 a continuación se presentan los tiempos necesitados en cada ejecución de los algoritmos; la tabla complementaria se encuentra en C.5 en el Apéndice C. Sólo se muestra un tiempo para cada instancia, ya que como se comentó anteriormente, primero se ejecuta MO-BEV y el criterio de paro de NSGA-RF es el tiempo utilizado por MO-BEV.

Se observa que al aumentar el número de iteraciones sin mejora del BEV-RFB en la fase II del MO-BEV sí hay un aumento muy considerable en el tiempo necesario para la ejecución. Con los parámetros (50,10) casi aumenta cinco veces más el tiempo de ejecución que con (20,5). Y cambiar los parámetros de (50,10) a (100,10) aumenta

Tabla 7.20: Comparación de resultados C(MO-BEV,NSGA-RF) y C(NSGA-RF,MO-BEV)

		Resultados MO-BEV			Resultados NSGA-RF		
n	Núm. Inst.	Núm. puntos frente Pareto	Núm. puntos domina–dos por NSGA-RF	C(NSGA-RF, MO-BEV)	Núm. puntos frente Pareto	Núm. puntos domina–dos por MO-BEV	C(MO-BEV, NSGA-RF)
42	1	49	0	0	47	1	0.02
	2	36	0	0	34	0	0
	3	28	0	0	31	2	0.06
	4	65	0	0	68	18	0.26
	5	50	0	0	50	0	0

casi el doble de tiempo, en la mayoría de los casos. Como era de esperarse, conforme aumenta el tamaño de la instancia, aumenta el tiempo de ejecución. Recordemos que el tiempo de ejecución de NSGA-RF es el mismo que utiliza MO-BEV, entonces esto explica que NSGA-RF mejora al ejecutarse por más tiempo.

Otro experimento interesante para comparar las dos metodologías es determinar, para una instancia dada, los puntos del frente de Pareto obtenido por MO-BEV que estén dominados por puntos el frente de Pareto obtenido por NSGA-RF y viceversa, de esta manera estaremos evaluando la calidad de las soluciones obtenidas. Para este propósito utilizaremos la métrica de Zitzler y Thiele [90] $C(A, B)$ que representa la proporción de puntos en la frontera estimada de B que son dominados por puntos eficientes en la frontera estimada de A. Los resultados con los parámetros (50,10) para las instancias más chicas se presentan en la tabla 7.20 y los resultados completos están en la tabla C.6 en el apéndice C.

Se observa que en la mayoría de las instancias el número de puntos obtenidos

por MO-BEV que no son dominados por puntos obtenidos por NSGA-RF es mayor.

Para terminar con los resultados obtenidos con el modelo biobjetivo, presentamos enseguida de manera gráfica unas comparaciones entre estos resultados para una instancia de 82 órdenes. En la figura 7.3.2 mostramos los diferentes frentes de Pareto obtenidos con el algoritmo MO-BEV y NSGA-RF con los diferentes parámetros para la instancia $\texttt{Instancia}82_1$. También se muestra la comparación entre los frentes obtenidos por ambas metodologías.

Más gráficas se presentan en el apéndice C.

7.3.3 Resumen

En resumen, de los resultados observados para el problema biobjetivo, podemos concluir que los dos algoritmos propuestos trabajan bien, logran obtener buenas aproximaciones a los frentes de Pareto en las instancias de prueba. NSGA-RF logra obtener buenos puntos no dominados pero MO-BEV logra mayor diversificación en la aproximación de la curva de eficiencia y en geneal de mayor calidad. Aumentar los parámetros de MO-BEV logra mejorar en la mayoría de los casos la aproximación del frente de Pareto pero con tiempos de cómputo mucho mayores, con los parámetros de (20,5) se logran buenos resultados con tiempos de cómputo razonables, sobre todo en instancias grandes.

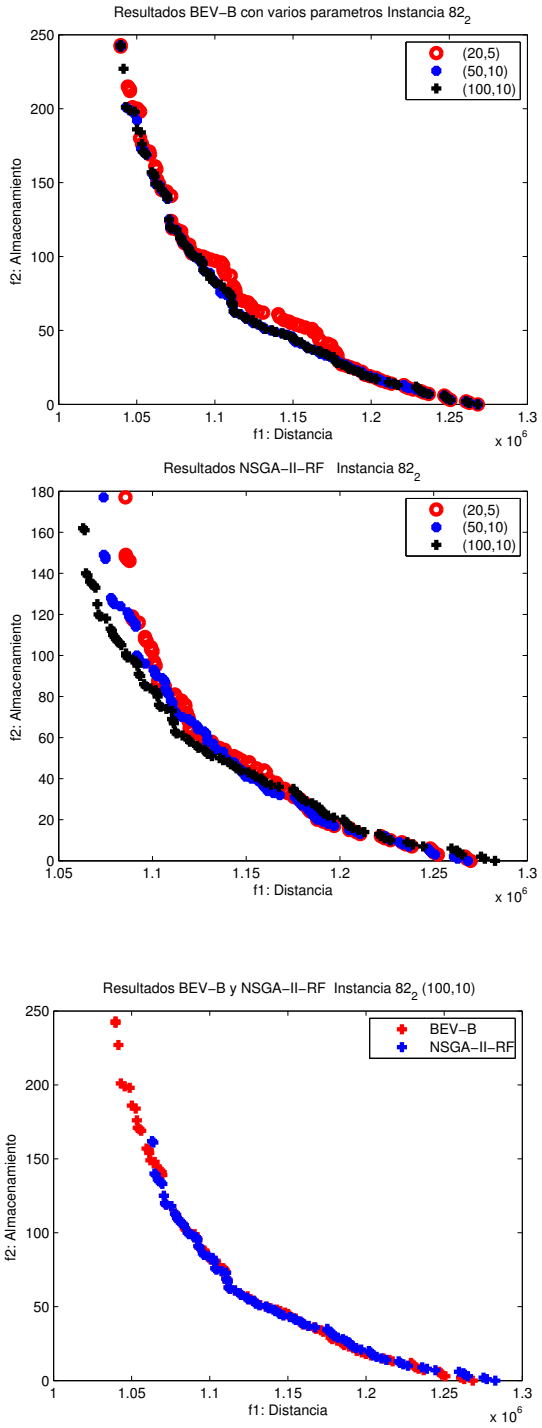


Figura 7.3: Frentes de Pareto obtenidos con MO-BEV y NSGA-RF en una instancia de 82 órdenes

Capítulo 8

Conclusiones y trabajo a futuro

El problema abordado en esta tesis estuvo inspirado en la problemática de una empresa distribuidora de productos, que deseaba disminuir los costos de envío a sus centros de distribución o delegaciones. Por consiguiente, deseaban diseñar las rutas de sus vehículos para la distribución de sus productos buscando una reducción en los costos de transportación.

Para resolver la problemática planteada se propuso permitir una cierta flexibilidad en la fecha de entrega, concretamente, permitir adelantar pedidos; esto es factible para la empresa ya que la entrega se hace a centros de distribución y no a usuarios finales. Como esta empresa en particular opera con productos perecederos, el adelanto permitido en la entrega fue de un día, sin embargo el modelo y las metodologías de solución desarrolladas consideraron el caso general de adelantar g días los pedidos.

Entonces el problema tratado fue diseñar de manera óptima las rutas de los vehículos que distribuyen los productos, satisfaciendo la demanda solicitada y realizando la entrega en la fecha inicialmente propuesta por las delegaciones o g días antes.

La política de adelantar pedidos genera un costo asociado al almacenamiento, lo cual repercute en la economía de la empresa, por lo que también se abordó el problema desde un enfoque biobjetivo buscando minimizar tanto el costo de transportación, medido por la distancia total recorrida por los vehículos, como el costo

del almacenamiento, medido por la cantidad de productos que es necesario almacenar. De acuerdo a la literatura revisada no encontramos antecedentes de trabajar problemas similares con este enfoque biobjetivo.

Entonces la investigación realizada en esta tesis comprende el estudio de la problemática inicial planteada por la empresa desde dos enfoques: mono-objetivo y multi-objetivo.

8.1 Conclusiones

Para el caso mono-objetivo se propusieron dos formulaciones matemáticas para modelar el problema considerando sólo el objetivo de minimizar la distancia total recorrida. Las dos formulaciones no sólo modelan el caso del adelanto un día, sino el caso general, por lo que estos modelos pueden ser aplicados a otros problemas donde sea posible aplicar la política de adelantar pedidos. La primera formulación controla el número total de rutas en todo el horizonte de planeación mas no el número de vehículos utilizados cada día. En este sentido la segunda formulación es más adecuada en general, sobre todo en problemas restringidos por la cantidad de vehículos.

Se desarrollaron dos metodologías de solución basadas en técnicas metaheurísticas. La primera está basada en un método híbrido de GRASP con encadenamiento de trayectorias (HGET) y la segunda en búsquedas por entornos variables BEV-RF). Se comprobó con diversos experimentos computacionales la eficiencia, robustez y precisión de los dos algoritmos propuestos para el problema mono-objetivo. Ambos logran llegar a la solución exacta en instancias pequeñas y al ser ejecutados en instancias pseudoreales obtuvieron en la mayoría de los casos soluciones mejores que las obtenidas por los otros algoritmos del estado del arte. Para realizar esta comparación fue necesario hacer los ajustes para modelar el problema como un PVRP.

HGET, en general, obtiene mejores resultados y requiere menor tiempo de

ejecución para lograr llegar a la mejor solución. El algoritmo BEV-RF también es preciso, requiere un poco más tiempo en lograr llegar a una buena solución al comparalo con HGET, pero logra buenos resultados. Cabe hacer mención que la forma de construir las vecindades en BEV-RF difiere de la forma clásica que utilizan la mayoría de los algoritmos basados en búsquedas por entornos variables para el VRP, donde usualmente se generalizan movimientos tipo r–opt. Se concluye que ambas metodologías son efectivas para resolver el problema planteado.

De la evaluación del desempeño de los diversos procedimientos incluidos en los algoritmos resaltan los relativos a la búsqueda local, la cual consiste esencialmente en movimientos de cadenas de vértices. Inicialmente se propusieron diferentes formas de realizar las inserciones y eliminaciones de estas cadenas de vértices, particularmente algunas basadas en la heurística de inserción generalizada (GENI) y las otras en los métodos más clásicos de inserción y eliminación. Los experimentos mostraron que al realizar los movimientos basados en GENI se consumía demasiado tiempo de computación y los resultados no mejoraban de forma notoria, por lo que recomendamos utilizar solamente las eliminaciones e inserciones clásicas, donde se obtienen buenos resultados más rápidamente.

Otra conclusión de la evaluación del desempeño de la búsqueda local fue la efectividad en el proceso de agilización de este procedimiento con lo que se denominó búsqueda local rápida y super rápida, en donde se dividen los vecindarios de la solución actual en subvecindarios más pequeños. Los resultados de los experimentos realizados muestran que ambas propuestas logran reducir considerablemente los tiempos utilizados en el proceso de la búsqueda local sin demeritar en la solución obtenida, sin embargo la búsqueda local super rápida require menos tiempo de ejecución. Esta estrategia puede aplicarse a problemas de ruteo o del agente viajero en general, ya que es eficaz y reduce de manera considerable los tiempos de ejecución.

Otro resultado interesante fue la evaluación de la fase del encadenamiento de trayectorias en HGET. De acuerdo a los experimentos realizados se observa que logra mejorar notablemente las soluciones que se obtienen sólo con la fase GRASP. Por

ello se recomienda la utilización de esta estrategia.

De acuerdo a los resultados de experimentos computacionales al resolver instancias pseudoreales, se observa que la política de adelanto de un día en las fechas de entrega de productos, provoca una disminución de la distancia total recorrida de alrededor del 20 %, al comparar con la forma en que actualmente la empresa realiza sus envíos. Esto representa una reducción considerable en el gasto de la compañía, por lo que consideramos que se ha cumplido satisfactoriamente uno de los primeros objetivos de la tesis, esto es, proporcionar a la empresa una estrategia para lograr disminuir sus costos.

En la segunda parte de la tesis se enfoca el problema desde una perspectiva más general, presentando una formulación matemática para modelar el problema desde el punto de vista biobjetivo considerando simultáneamente los objetivos de minimizar la distancia total recorrida y el almacenamiento generado.

Del trabajo desarrollado con este enfoque multi-objetivo se concluye que ambos objetivos se encuentran en conflicto, disminuye uno al tiempo que el otro aumenta y viceversa.

Se diseñaron dos metodologías de solución al problema biobjetivo, basadas en técnicas metaheurísticas. La primera se puede ver como una generalización del método por entornos variables diseñada para el problema con un solo objetivo, pero insertada dentro del esquema MOAMP. Esto es, se realiza una serie de búsquedas por entornos variables enlazadas, por lo que denotamos el procedimiento como MO-BEV. La segunda metodología es un método evolutivo basado en NSGA-II, dotado de mecanismos de cruce y mutación desarrollados específicamente para el problema. Esto es, operadores enfocados directamente a los objetivos de minimizar la distancia total recorrida por las rutas y el almacenamiento ocasionado por la flexibilidad en las fechas de entrega. Lo denotamos NSGA-RF.

Esta segunda parte del trabajo permite dar cumplimiento a los objetivos que nos planteamos en cuanto a estudiar el problema desde una perspectiva más general,

que permitieran su aplicación en situaciones donde el adelanto en la entrega pudiera ser mayor.

Ambos algoritmos fueron implementados computacionalmente y se comprobó con diversos experimentos computacionales su eficiencia, robustez y precisión. Las dos metodologías mostraron ser competitivas, logrando aproximar de forma densa y con calidad la frontera de Pareto. De acuerdo a experimentos realizados, en general, MO-BEV obtiene puntos de más calidad que NSGA-RF.

Como conclusión general podemos mencionar que se logró el objetivo inicialmente propuesto en el desarrollo de esta tesis, esto es, proponerle a la compañía una alternativa de operación que permita disminuir los costos de transportación permitiendo el adelanto en las fechas de entrega de productos. Los modelos y metodologías propuestos pueden aplicarse a problemas similares donde se permita flexibilidad en las fechas de entrega. Adicionalmente las metodologías de solución pueden adaptarse a problemas de ruteo con otras características.

Asimismo, al estudiar el problema desde la perspectiva biobjetivo, con los algoritmos propuestos se logra construir un conjunto de soluciones compromiso que permitirán al tomador de decisiones escoger una adecuada a sus preferencias.

Los objetivos inicialmente planteados en el desarrollo de esta tesis fueron cubiertos en su totalidad.

8.2 Trabajo a futuro

En el desarrollo de esta tesis se ha considerado en todo momento que las demandas de un centro de distribución deben ser satisfechas por un solo vehículo, esto es, no se ha permitido particionar pedidos. Esto ocurre comunmente en la práctica y el motivo es que al particionar pedidos se incurre en costos adicionales al descargar los productos porque los operarios tienen que ser convocados dos o más veces. El particionar pedidos fue una de las estrategias propuestas inicialmente a la empre-

sa repostera, pero la compañía de momento no podía absorber el gasto extra de operarios, por lo que no se consideró esta opción.

Sin embargo, una generalización del problema estudiado sería permitir particionar los pedidos y medir el impacto en los costos de la empresa. Se puede incluir este costo como un objetivo mas del modelo y trabajar el problema de tres objetivos tratando de generalizar las metodologías propuestas.

Otro estudio que puede resultar interesante es respecto a las metodologías de solución: su aplicación en otro tipo de problemas y el desarrollo de otras metodologías basadas en las aquí propuestas. Particularmente el procedimiento del encadeamiento de trayectorias y la búsqueda por entornos variables pueden aplicarse a otros problemas de ruteo de vehículos, consideramos que pueden obtener buenos resultados.

Apéndice A

GENI: Heurística de inserción generalizada

La Heurística de inserción generalizada GENI, desarrollada por Gendreau et al. [41], es una de las heurísticas más efectivas reportadas en la literatura para el TSP. La describimos brevemente enseguida.

GENI es una heurística constructiva para el problema del agente viajero, parte de un tour con 3 vértices y va insertando un vértice a la vez hasta completar todos los nodos. GENI difiere del algoritmo de inserción clásico, ya que la inserción de un vértice v en el tour no se hace necesariamente entre dos vértices consecutivos, sin embargo, después de la inserción, estos dos vértices serán adyacentes a v. La forma de operar del algoritmo es la siguiente:

Supongamos que tenemos un tour parcial formado por algunos vértices y que queremos insertar v entre dos vértices cualesquiera v_i y v_j del tour. Para una orientación dada, sea v_k un vértice en la ruta de v_j a v_i, y v_l un vértice en la ruta de v_i a v_j. Denotaremos para un vértice v_h cualquiera como v_{h-1} su predecesor en el tour y v_{h+1} su sucesor dentro del tour. El algoritmo maneja dos tipos de inserciones llamadas tipo I y II, las cuales comentamos enseguida.

Inserción tipo I

Se considera que $v_k \neq v_i$ y $v_k \neq v_j$. Al insertar v eliminamos los arcos (v_i, v_{i+1}), (v_j, v_{j+1}) y (v_k, v_{k+1}) y se reemplazan por (v_i, v), (v, v_j), (v_{i+1}, v_k) y (v_{j+1}, v_{k+1}). Al hacerlo hay que cambiar la orientación del tour en $(v_{i+1}, \ldots, v_j)$ y $(v_{j+1}, \ldots v_k)$. Geométricamente tenemos lo que se ilustra en la figura A.1.

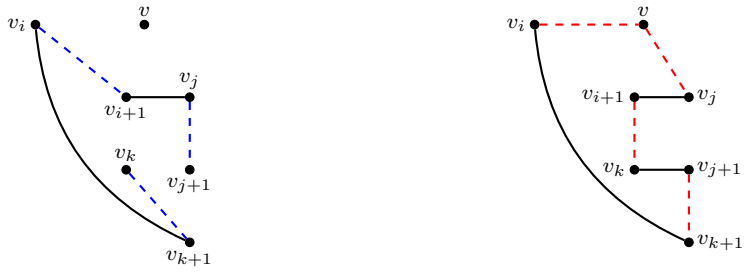


Figura A.1: Movimiento Tipo GENI-I

Si consideramos $j = i + 1$ y $k = j + 1$ obtenemos el procedimiento de inserción clásico. Se consideran las dos posibles orientaciones del tour para cada posible inserción.

Inserción tipo II

Se considera $v_k \neq v_j$ y $v_k \neq v_{j+1}$; $v_l \neq v_i$ y $v_l \neq v_{i+1}$. Al insertar v en el tour se eliminan los arcos (v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1}) y (v_{k-1}, v_k). Estos arcos se reemplazan por (v_i, v), (v, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) y (v_{i+1}, v_k). Hay que cambiar la orientación del tour en $(v_{i+1}, \ldots, v_{l-1})$ y $(v_l, \ldots v_j)$. Lo ilustramos en la figura A.2.

Al igual que con el tipo I, se consideran las dos posibles orientaciones del tour.

Como el número posible de elecciones de v_i, v_j, v_k, v_l es del orden de n^4, donde n es número de vértices en el tour actual, se limita la búsqueda como sigue. Para cualquier vértice v se define su p–vecindad $N_p(v)$ como el conjunto de p vértices del tour más cercanos a v, esto en términos de la matriz de costos C. Si v tiene menos de p vecinos en el tour, todos éstos pertenecen a $N_p(v)$. Para un parámetro

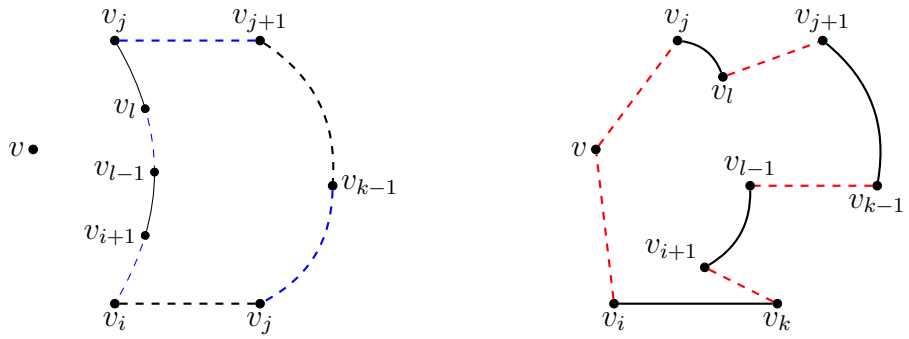


Figura A.2: Movimiento tipo GENI-II

p dado se selecciona primero $v_i, v_j \in N_p(v)$, $v_k \in N_p(v_{i+1})$ y $v_l \in N_p(v_{j+1})$, también se consieran todas las inseciones para v entre dos vértices consecutivos v_i y v_{i+1} para $v_i \in N_p(v)$, con esto tenemos del orden de p^4 elecciones para v_i, v_j, v_k y v_l, en la práctica p es un número relativamente pequeño, lo cual reduce el tamaño de la búsqueda.

El algoritmo GENI para el TSP es el siguiente:

1. Crear un tour inicial seleccionando un subconjunto arbitrario de 3 vértices.

   a) Inicializar las p–vecindades de todos los vértices.

2. Seleccionar arbitrariamente un vértice v que no esté en el tour.

   a) Implementar la inserción de menor costo considerando los dos tipos de movimientos y las dos orientaciones del tour.

   b) Actualizar las p–vecindades de todos los vértices tomando en cuenta que v pertenece al tour.

3. Si todos los vértices pertenecen al tour, parar. Si no, ir al paso 2.

Al hacer la implementación de la inserción se consideran todas las posibles elecciones de v_i, v_j, v_k y v_l que pertenezcan a las p–vecindades requeridas y se selecciona la de menor costo.

Apéndice B

Resultados computacionales modelo mono-objetivo

Tabla B.1: Resultados GRASP: Comparación de resultados con búsqueda local completa y búsqueda local rápida

				Búsqueda Local Completa			Búsqueda Local Rápida		
n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.591	0.090	111414	0.240	0.050
42	15	1	1	90934	3.625	3.475	90934	1.582	1.512
42	15	1	2	88874	4.867	2.043	88874	2.264	0.992
42	15	1	3	82973	10.565	0.871	82973	4.236	0.370
42	15	1	4	80473	17.595	2.874	80473	7.211	1.272
42	15	2	0	112218	1.181	0.250	112218	0.501	0.121
42	15	2	1	93439	7.020	5.678	93439	2.924	2.414
42	15	2	2	86529	16.204	2.063	86529	6.589	1.011
42	15	2	3	82344	33.238	9.884	82344	13.730	4.647
42	15	2	4	80011	42.691	0.480	80011	17.846	0.250
42	15	3	0	111547	0.841	0.180	111547	0.370	0.090
42	15	3	1	90763	4.457	0.091	90763	1.913	0.050

Continúa en la siguiente página ...

Tabla B.1 ... Continuación de la página anterior

				Búsqueda Local Completa			Búsqueda Local Rápida		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Inst.	g	costo	Total	mejor sol.	costo	Total	mejor sol.
42	15	3	2	84838	7.711	0.240	84838	3.324	0.160
42	15	3	3	81690	13.620	2.895	81690	5.238	1.172
42	15	3	4	78942	13.589	8.071	78942	5.658	3.495
42	15	4	0	116914	0.130	0.130	116914	0.051	0.051
42	15	4	1	98202	0.261	0.181	98202	0.110	0.070
42	15	4	2	90539	1.382	0.410	90539	0.591	0.180
42	15	4	3	90577	1.592	0.520	90577	0.711	0.230
42	15	4	4	86031	5.117	0.481	86031	1.853	0.191
42	15	5	0	112494	0.290	0.140	112494	0.141	0.081
42	15	5	1	92730	4.156	0.080	92730	1.642	0.040
42	15	5	2	84291	11.466	1.122	84291	4.777	0.541
42	15	5	3	80146	14.431	5.598	80146	5.808	2.454
42	15	5	4	77221	18.247	0.631	77221	7.390	0.260
42	22	1	0	129292	3.035	0.311	129292	1.091	0.130
42	22	1	1	107855	6.399	3.155	107855	2.113	1.072
42	22	1	2	104007	11.136	2.794	104007	3.496	0.992
42	22	1	3	100300	14.480	1.231	100300	4.347	0.451
42	22	1	4	99929	15.922	0.200	99929	4.827	0.091
42	22	2	0	130328	3.425	0.221	130328	1.242	0.090
42	22	2	1	116876	10.595	0.090	116876	3.936	0.040
42	22	2	2	108268	16.063	0.461	108268	5.919	0.200
42	22	2	3	103889	22.622	1.532	103889	8.202	0.591
42	22	2	4	102822	23.774	0.371	102822	8.793	0.170
42	22	3	0	133664	0.991	0.170	133664	0.341	0.070
42	22	3	1	117340	6.159	0.440	117340	2.113	0.180

Continúa en la siguiente página ...

Tabla B.1 ... Continuación de la página anterior

				Búsqueda Local Completa			Búsqueda Local Rápida		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Inst.	g	costo	Total	mejor sol.	costo	Total	mejor sol.
42	22	3	2	110464	12.978	4.346	110464	4.527	1.522
42	22	3	3	107117	14.390	4.897	107117	4.918	1.773
42	22	3	4	107117	17.395	5.427	107117	6.178	1.992
42	22	4	0	133609	0.301	0.071	133609	0.150	0.040
42	22	4	1	122655	2.494	0.681	122655	0.941	0.270
42	22	4	2	122103	3.074	0.901	122103	1.042	0.321
42	22	4	3	116109	6.479	0.461	116109	2.133	0.171
42	22	4	4	116202	7.972	0.952	116202	2.563	0.330
42	22	5	0	135173	3.786	0.121	135173	1.182	0.050
42	22	5	1	111888	9.744	2.293	111888	3.495	0.961
42	22	5	2	108869	13.028	0.390	108869	4.607	0.181
42	22	5	3	106522	20.459	5.758	106522	7.351	2.224
42	22	5	4	102978	32.377	0.170	102978	12.097	0.070
102	15	1	0	226289	88.998	38.275	226289	17.695	7.811
102	15	1	1	185606	488.333	22.663	185606	93.614	5.337
102	15	1	2	175820	678.946	59.446	175820	151.298	13.590
102	15	1	3	165467	635.143	74.828	165467	139.721	18.918
102	15	1	4	161443	1000.749	5.078	161443	219.175	1.312
104	15	2	0	213084	274.816	18.437	213084	50.883	4.156
104	15	2	1	179176	615.264	198.405	179176	122.577	41.971
104	15	2	2	167964	1065.261	13.449	167964	197.504	2.915
104	15	2	3	158295	1288.593	162.975	158295	262.187	41.300
104	15	2	4	155936	1816.943	104.060	155936	373.717	26.969
104	15	3	0	228584	122.667	26.098	228584	22.532	5.177
104	15	3	1	195522	309.736	273.273	195522	59.305	53.476

Continúa en la siguiente página ...

Tabla B.1 ... Continuación de la página anterior

				Búsqueda Local Completa			Búsqueda Local Rápida		
n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.	costo	Tiempo Total	Tiempo mejor sol.
104	15	3	2	185088	420.585	113.793	185088	86.394	25.797
104	15	3	3	174832	539.345	152.619	174832	115.426	37.755
104	15	3	4	168583	800.612	43.843	168583	166.209	9.503
105	15	4	0	222838	253.724	31.405	222838	49.762	6.610
105	15	4	1	189931	628.544	4.907	189931	109.027	1.232
96	15	5	0	222998	32.456	22.763	222998	6.600	4.677
96	15	5	1	195544	167.791	13.840	195544	33.178	2.944
101	22	1	0	280250	260.435	10.616	280250	39.206	1.903
101	22	1	1	248824	508.562	46.597	248824	72.834	7.590
100	22	2	0	304105	70.311	10.255	304105	13.209	2.634
100	22	2	1	275585	219.466	97.641	275585	35.541	16.684
97	22	3	0	273673	67.838	17.205	273673	12.117	3.495
97	22	3	1	246640	189.503	11.156	246640	32.306	2.323
109	22	4	0	312437	285.791	186.088	312437	41.770	28.020
109	22	4	1	280172	638.017	19.448	280172	93.064	3.355
106	22	5	0	288365	305.079	90.911	288365	45.105	14.441
106	22	5	1	263434	560.986	82.458	263434	80.686	13.800
183	15	1	0	354273	3971.040	3397.996	354273	456.456	392.534
183	15	1	1	305670	5402.188	83.200	305670	675.171	11.737
184	15	2	0	354214	5466.661	1907.052	354214	799.609	294.663
184	15	2	1	305709	8432.235	210.272	305709	1030.072	26.168
188	15	3	0	370107	4810.677	132.340	370107	606.001	15.733
188	15	3	1	320710	6712.883	653.911	320710	842.712	89.609
186	15	4	0	360487	5370.813	1880.814	360487	676.613	242.709
186	15	4	1	319238	8273.897	1367.376	319238	1041.227	188.261

Continúa en la siguiente página ...

Tabla B.1 ... Continuación de la página anterior

				Búsqueda Local Completa			Búsqueda Local Rápida		
n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.	costo	Tiempo Total	Tiempo mejor sol.
185	15	5	0	366519	4524.957	110.289	366519	533.697	15.001
185	15	5	1	319341	5942.024	1484.725	319341	781.734	205.786
185	22	1	0	489251	3481.175	132.410	489251	327.972	14.982
185	22	1	1	432468	4353.370	334.431	432468	431.140	35.581
192	22	2	0	546288	1097.998	759.602	546288	96.489	68.188
192	22	2	1	495186	2402.575	834.130	495186	218.744	77.862
184	22	3	0	480545	5697.503	5346.919	480545	544.142	513.118
184	22	3	1	436617	6946.990	83.551	436617	698.915	10.725
183	22	4	0	481332	6088.324	4470.067	481332	557.883	421.697
183	22	4	1	434577	8823.896	4273.545	434577	963.145	469.946
185	22	5	0	478531	3320.845	136.757	478531	298.840	16.474
185	22	5	1	438291	4177.156	519.187	438291	385.044	51.274

Tabla B.2: Resultados GRASP sólo con la fase constructiva

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	113202	0.031	0.031
42	15	1	1	95403	0.031	0.015
42	15	1	2	90850	0.031	0.016
42	15	1	3	85794	0.016	0.016
42	15	1	4	81421	0.015	0.000
42	15	2	0	118221	0.031	0.031
42	15	2	1	94977	0.032	0.016

Continúa en la siguiente página ...

Tabla B.2 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
42	15	2	2	88444	0.032	0.000
42	15	2	3	84595	0.031	0.000
42	15	2	4	81794	0.047	0.015
42	15	3	0	115136	0.015	0.015
42	15	3	1	93265	0.032	0.016
42	15	3	2	88300	0.031	0.015
42	15	3	3	85183	0.031	0.000
42	15	3	4	81565	0.031	0.000
42	15	4	0	120979	0.031	0.016
42	15	4	1	100672	0.016	0.016
42	15	4	2	95288	0.016	0.016
42	15	4	3	95463	0.015	0.015
42	15	4	4	92062	0.016	0.016
42	15	5	0	120023	0.031	0.015
42	15	5	1	93972	0.031	0.000
42	15	5	2	85775	0.032	0.000
42	15	5	3	81690	0.031	0.000
42	15	5	4	79327	0.031	0.000
42	22	1	0	129963	0.031	0.016
42	22	1	1	110012	0.047	0.016
42	22	1	2	107152	0.047	0.016
42	22	1	3	103092	0.031	0.000
42	22	1	4	102099	0.047	0.016
42	22	2	0	135835	0.063	0.031
42	22	2	1	118809	0.047	0.000
42	22	2	2	112015	0.062	0.031

Continúa en la siguiente página ...

Tabla B.2 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
42	22	2	3	108942	0.063	0.031
42	22	2	4	106100	0.062	0.015
42	22	3	0	138363	0.031	0.031
42	22	3	1	120425	0.047	0.032
42	22	3	2	114312	0.047	0.016
42	22	3	3	109035	0.047	0.016
42	22	3	4	109703	0.047	0.016
42	22	4	0	134129	0.047	0.031
42	22	4	1	124319	0.031	0.015
42	22	4	2	124826	0.031	0.031
42	22	4	3	118840	0.047	0.031
42	22	4	4	118691	0.031	0.016
42	22	5	0	136792	0.063	0.047
42	22	5	1	113842	0.062	0.031
42	22	5	2	109706	0.063	0.016
42	22	5	3	109791	0.062	0.015
42	22	5	4	104604	0.063	0.016
102	15	1	0	238743	0.265	0.140
102	15	1	1	195079	0.297	0.079
104	15	2	0	221612	0.360	0.110
104	15	2	1	186512	0.328	0.047
104	15	3	0	236909	0.234	0.187
104	15	3	1	203555	0.250	0.125
105	15	4	0	234523	0.313	0.156
105	15	4	1	194540	0.328	0.047
96	15	5	0	235788	0.156	0.094

Continúa en la siguiente página ...

Tabla B.2 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
96	15	5	1	200155	0.172	0.078
101	22	1	0	291491	0.437	0.219
101	22	1	1	257592	0.438	0.110
100	22	2	0	312003	0.391	0.250
100	22	2	1	285004	0.375	0.187
97	22	3	0	280058	0.390	0.218
97	22	3	1	251902	0.391	0.156
109	22	4	0	321049	0.609	0.234
109	22	4	1	285722	0.594	0.094
106	22	5	0	294745	0.547	0.203
106	22	5	1	271240	0.547	0.219
183	15	1	0	371941	1.812	0.843
183	15	1	1	315499	1.781	0.360
184	15	2	0	380945	2.110	0.797
184	15	2	1	321824	2.047	0.547
188	15	3	0	394690	2.078	0.937
188	15	3	1	331075	2.031	0.297
186	15	4	0	387026	2.219	0.719
186	15	4	1	326088	2.172	0.344
185	15	5	0	392803	2.031	0.687
185	15	5	1	328545	1.969	0.266
185	22	1	0	520116	3.438	1.594
185	22	1	1	445585	3.344	1.234
192	22	2	0	583114	3.406	2.203
192	22	2	1	506244	3.329	1.407
184	22	3	0	512322	3.843	1.703

Continúa en la siguiente página ...

Tabla B.2 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
184	22	3	1	450873	3.766	1.125
183	22	4	0	525578	4.312	1.859
183	22	4	1	454448	4.219	0.797
185	22	5	0	499481	3.453	1.453
185	22	5	1	451413	3.360	1.407

Tabla B.3: Resultados GRASP: Comparación fijando en 3 el tamaño de la cadena de vértices a mover en la búsqueda local

				Tamaño cadena libre			Tamaño cadena 3		
n	$Q_{\text{máx}}$	Núm. Inst.	F	costo	Tiempo Total	Tiempo mejor sol.	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.172	0.031	111414	0.156	0.031
42	15	1	1	90934	1.125	1.078	90934	1.015	0.969
42	15	1	2	88874	1.610	0.704	88874	1.453	0.625
42	15	1	3	82973	2.906	0.266	82973	2.719	0.235
42	15	1	4	80473	4.953	0.891	80473	4.703	0.828
42	15	2	0	112218	0.360	0.078	112218	0.328	0.078
42	15	2	1	93439	2.062	1.718	93439	1.860	1.531
42	15	2	2	86529	4.625	0.703	86529	4.125	0.593
42	15	2	3	82344	9.718	3.312	82344	8.328	2.766
42	15	2	4	80011	12.719	0.172	80011	10.985	0.141
42	15	3	0	111547	0.281	0.078	111547	0.219	0.047
42	15	3	1	90763	1.375	0.031	90763	1.187	0.046
42	15	3	2	84838	2.328	0.109	84838	2.047	0.094

Continúa en la siguiente página ...

Tabla B.3 ... Continuación de la página anterior

				Tamaño cadena libre			Tamaño cadena 3		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Inst.	F	costo	Total	mejor sol.	costo	Total	mejor sol.
42	15	3	3	81690	3.703	0.844	81690	3.328	0.735
42	15	3	4	78942	4.047	2.516	78942	3.594	2.203
42	15	4	1	98202	0.093	0.062	98202	0.078	0.047
42	15	4	2	90539	0.422	0.125	90539	0.422	0.125
42	15	4	3	90577	0.485	0.156	90577	0.468	0.140
42	15	4	4	86031	1.297	0.141	86031	1.250	0.125
42	15	5	0	112494	0.094	0.047	112494	0.109	0.062
42	15	5	1	92730	1.172	0.032	92730	1.063	0.031
42	15	5	2	84291	3.375	0.390	84291	3.031	0.328
42	15	5	3	80146	4.016	1.703	80146	3.687	1.562
42	15	5	4	77221	5.156	0.188	77221	4.813	0.172
42	22	1	0	129292	0.812	0.094	129292	0.750	0.094
42	22	1	1	107855	1.547	0.797	107855	1.485	0.750
42	22	1	2	104007	2.468	0.703	104007	2.360	0.672
42	22	1	3	100300	3.093	0.328	100300	2.985	0.313
42	22	1	4	99929	3.453	0.062	99929	3.359	0.062
42	22	2	0	130328	0.891	0.063	130328	0.844	0.062
42	22	2	1	116876	2.843	0.031	116876	2.563	0.016
42	22	2	2	108268	4.187	0.141	108268	3.985	0.141
42	22	2	3	103889	5.890	0.437	103889	5.391	0.391
42	22	2	4	102822	6.125	0.125	102822	5.938	0.109
42	22	3	0	133664	0.265	0.046	133664	0.266	0.047
42	22	3	1	117340	1.594	0.140	117340	1.515	0.140
42	22	3	2	110464	3.344	1.141	110464	3.016	1.047
42	22	3	3	107117	3.531	1.250	107117	3.344	1.187

Continúa en la siguiente página ...

Tabla B.3 ... Continuación de la página anterior

				Tamaño cadena libre			Tamaño cadena 3		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Inst.	F	costo	Total	mejor sol.	costo	Total	mejor sol.
42	22	3	4	107117	4.453	1.437	107117	4.156	1.359
42	22	4	0	133609	0.109	0.031	133609	0.110	0.032
42	22	4	1	122655	0.687	0.203	122655	0.672	0.188
42	22	4	2	122103	0.734	0.219	122103	0.719	0.219
42	22	4	3	116109	1.516	0.125	116109	1.453	0.109
42	22	4	4	116202	1.828	0.250	116202	1.828	0.234
42	22	5	0	135173	0.906	0.047	135173	0.860	0.047
42	22	5	1	111888	2.609	0.719	111888	2.391	0.656
42	22	5	2	108869	3.281	0.140	108869	2.984	0.109
42	22	5	3	106522	5.266	1.579	106522	4.672	1.422
42	22	5	4	102978	8.641	0.078	102978	7.609	0.046
102	15	1	0	226289	12.766	5.656	226289	10.812	4.812
102	15	1	1	185606	67.125	3.906	185606	56.797	3.156
104	15	2	0	213084	35.828	2.891	213084	31.406	2.485
104	15	2	1	179176	85.079	29.235	179176	71.578	24.171
104	15	3	0	228584	15.609	3.547	228584	13.719	3.078
104	15	3	1	195522	39.156	35.062	195522	34.313	30.703
105	15	4	0	222838	32.718	4.437	222949	27.532	3.688
105	15	4	1	189931	76.718	0.859	189983	64.391	0.703
96	15	5	0	222998	4.703	3.344	222998	4.250	3.031
96	15	5	1	195544	23.422	2.031	195544	21.250	1.844
101	22	1	0	280250	28.281	1.391	280250	25.891	1.266
101	22	1	1	248824	51.984	5.453	248824	48.688	5.063
100	22	2	0	304105	8.578	1.562	304105	7.812	1.390
100	22	2	1	275585	27.282	13.813	275585	23.703	10.890

Continúa en la siguiente página ...

Tabla B.3 ... Continuación de la página anterior

				Tamaño cadena libre			Tamaño cadena 3		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Inst.	F	costo	Total	mejor sol.	costo	Total	mejor sol.
97	22	3	0	273673	8.750	2.531	273673	8.093	2.359
97	22	3	1	246640	23.047	1.657	246640	21.516	1.547
109	22	4	0	312437	29.812	20.062	312437	27.594	18.516
109	22	4	1	280172	65.531	2.391	280172	62.610	2.250
106	22	5	0	288365	32.265	10.344	288365	28.657	9.172
106	22	5	1	263434	56.062	9.547	263434	53.360	8.969
183	15	1	0	354764	34.813	25.469	354764	27.578	20.328
183	15	1	1	312384	37.922	3.672	312384	32.390	3.109
184	15	2	0	356596	42.563	22.578	356596	35.109	18.703
184	15	2	1	313319	49.328	41.360	313319	41.328	34.703
188	15	3	0	373753	41.797	7.704	373753	33.750	6.500
188	15	3	1	324686	39.938	7.625	325689	34.484	9.812
186	15	4	0	364368	40.453	40.453	364368	32.922	32.922
186	15	4	1	324335	48.328	25.828	324335	39.891	21.282
185	15	5	0	369444	39.844	24.953	369444	32.359	20.234
185	15	5	1	322077	44.344	27.859	322077	36.422	22.765
185	22	1	0	489407	20.922	9.375	489407	18.828	8.375
185	22	1	1	442640	17.656	7.718	442640	16.594	7.328
192	22	2	0	547884	18.109	7.828	547884	16.906	7.281
192	22	2	1	503761	15.703	14.219	503761	15.063	13.657
184	22	3	0	483635	33.406	4.968	483635	25.531	3.719
184	22	3	1	440671	28.125	12.719	441986	24.234	8.359
183	22	4	0	485025	34.094	4.610	485025	30.250	4.125
183	22	4	1	440740	31.578	19.110	440740	28.532	17.344
185	22	5	0	481267	22.766	22.766	481267	20.312	20.312

Continúa en la siguiente página ...

Tabla B.3 ... Continuación de la página anterior

				Tamaño cadena libre			Tamaño cadena 3		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Inst.	F	costo	Total	mejor sol.	costo	Total	mejor sol.
185	22	5	1	441552	17.922	17.141	441552	16.531	15.781

Tabla B.4: Resultados GRASP sin la heurística GENI en la búsqueda local

		Núm.			Tiempo	Tiempo
n	$Q_{\text{máx}}$	Inst.	F	costo	Total	mejor sol.
42	15	1	0	111414	0.172	0.032
42	15	1	1	90934	1.110	1.063
42	15	1	2	88874	1.531	0.672
42	15	1	3	82973	2.656	0.234
42	15	1	4	80473	4.844	0.813
42	15	2	0	112218	0.359	0.078
42	15	2	1	93439	1.657	1.407
42	15	2	2	86529	3.890	0.609
42	15	2	3	82344	7.735	0.297
42	15	2	4	80011	10.031	0.140
42	15	3	0	111547	0.219	0.047
42	15	3	1	90763	1.250	0.031
42	15	3	2	84838	2.141	0.110
42	15	3	3	81690	3.172	0.719
42	15	3	4	78942	3.406	1.015
42	15	4	1	98202	0.078	0.046
42	15	4	2	90539	0.422	0.125
42	15	4	3	92409	0.468	0.312

Continúa en la siguiente página ...

Tabla B.4 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
42	15	4	4	86031	1.250	0.141
42	15	5	0	112672	0.094	0.047
42	15	5	1	92730	1.188	0.031
42	15	5	2	84291	3.140	0.359
42	15	5	3	80146	3.735	1.547
42	15	5	4	77221	4.859	0.343
42	22	1	0	129292	0.812	0.109
42	22	1	1	107855	1.579	0.797
42	22	1	2	104007	2.468	0.687
42	22	1	3	100300	3.094	0.328
42	22	1	4	99929	3.328	0.047
42	22	2	0	130328	0.922	0.062
42	22	2	1	116876	2.609	0.031
42	22	2	2	108268	4.375	0.141
42	22	2	3	103889	5.844	0.406
42	22	2	4	102822	5.969	0.125
42	22	3	0	133664	0.250	0.046
42	22	3	1	117340	1.593	0.140
42	22	3	2	110464	3.188	1.047
42	22	3	3	107117	3.625	1.281
42	22	3	4	107117	4.297	1.422
42	22	4	0	133609	0.109	0.031
42	22	4	1	122655	0.688	0.203
42	22	4	2	122103	0.765	0.234
42	22	4	3	116109	1.578	0.125
42	22	4	4	116202	1.844	0.250

Continúa en la siguiente página ...

Tabla B.4 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
42	22	5	0	135173	0.906	0.031
42	22	5	1	111888	2.547	0.687
42	22	5	2	108869	3.172	0.125
42	22	5	3	106522	4.953	1.468
42	22	5	4	102978	8.328	0.047
102	15	1	0	226289	10.187	4.562
102	15	1	1	185606	44.204	2.438
104	15	2	0	213084	29.344	2.469
104	15	2	1	179176	61.703	21.281
104	15	3	0	229093	13.188	10.672
104	15	3	1	195523	31.734	28.328
105	15	4	0	222838	27.062	3.672
105	15	4	1	189931	56.610	0.657
96	15	5	0	222998	4.187	2.953
96	15	5	1	195724	18.000	1.641
101	22	1	0	280250	24.094	1.250
101	22	1	1	248824	43.844	4.719
100	22	2	0	304105	7.281	1.344
100	22	2	1	275585	21.188	9.953
97	22	3	0	273673	7.859	2.281
97	22	3	1	246640	19.625	1.437
109	22	4	0	312437	25.687	17.406
109	22	4	1	280172	55.407	2.110
106	22	5	0	288365	25.562	8.312
106	22	5	1	263434	46.266	7.985
183	15	1	0	355262	25.078	18.375

Continúa en la siguiente página ...

Tabla B.4 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
183	15	1	1	313473	27.250	2.594
184	15	2	0	357438	31.422	1.485
184	15	2	1	314952	35.485	26.438
188	15	3	0	373753	32.625	6.171
188	15	3	1	324686	30.375	5.531
186	15	4	0	364490	30.593	30.593
186	15	4	1	324335	35.672	18.985
185	15	5	0	369152	31.485	1.406
185	15	5	1	323259	31.468	31.468
185	22	1	0	489407	17.782	7.907
185	22	1	1	442640	14.796	6.484
192	22	2	0	547884	15.594	6.734
192	22	2	1	503761	13.469	12.203
184	22	3	0	482586	22.078	3.281
184	22	3	1	440671	20.265	8.937
183	22	4	0	485025	28.016	3.672
183	22	4	1	440740	25.625	15.468
185	22	5	0	481267	19.140	19.140
185	22	5	1	441552	14.875	14.219

Tabla B.5: Resultados GRASP sin cambio de orientación y sin GENI

n	$Q_{\text{máx}}$	Núm. Inst.	F	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.047	0.031
42	15	1	1	90934	0.203	0.203
42	15	1	2	88874	0.297	0.141
42	15	1	3	82973	0.453	0.062
42	15	1	4	80605	0.734	0.031
42	15	2	0	112218	0.094	0.032
42	15	2	1	93439	0.328	0.281
42	15	2	2	86529	0.703	0.125
42	15	2	3	82344	1.203	0.063
42	15	2	4	80011	1.454	0.032
42	15	3	0	111547	0.062	0.031
42	15	3	1	90763	0.234	0.000
42	15	3	2	84838	0.407	0.188
42	15	3	3	81690	0.562	0.140
42	15	3	4	79074	0.531	0.172
42	15	4	1	98202	0.016	0.016
42	15	4	2	90539	0.093	0.047
42	15	4	3	92409	0.094	0.063
42	15	4	4	86031	0.219	0.031
42	15	5	0	112672	0.031	0.016
42	15	5	1	92137	0.235	0.219
42	15	5	2	84291	0.578	0.078
42	15	5	3	80146	0.625	0.250
42	15	5	4	77221	0.828	0.062

Continúa en la siguiente página ...

Tabla B.5 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
42	22	1	0	129292	0.172	0.031
42	22	1	1	107855	0.343	0.187
42	22	1	2	104007	0.454	0.141
42	22	1	3	100300	0.546	0.062
42	22	1	4	99929	0.579	0.016
42	22	2	0	130328	0.203	0.015
42	22	2	1	116876	0.500	0.015
42	22	2	2	108268	0.781	0.031
42	22	2	3	103889	0.984	0.078
42	22	2	4	102822	1.000	0.032
42	22	3	0	133664	0.078	0.032
42	22	3	1	117340	0.329	0.032
42	22	3	2	110464	0.562	0.187
42	22	3	3	107117	0.641	0.234
42	22	3	4	107117	0.750	0.250
42	22	4	0	133609	0.047	0.031
42	22	4	1	122655	0.156	0.062
42	22	4	2	122103	0.156	0.062
42	22	4	3	116109	0.297	0.047
42	22	4	4	116202	0.328	0.062
42	22	5	0	135173	0.218	0.015
42	22	5	1	111888	0.516	0.157
42	22	5	2	108869	0.625	0.047
42	22	5	3	106204	0.938	0.328
42	22	5	4	102978	1.359	0.015
102	15	1	0	226289	2.391	1.156

Continúa en la siguiente página ...

Tabla B.5 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
102	15	1	1	185662	9.265	4.218
104	15	2	0	213084	6.938	0.688
104	15	2	1	179176	12.578	4.422
104	15	3	0	229559	2.937	1.187
104	15	3	1	195523	6.672	5.938
105	15	4	0	223308	5.875	0.891
105	15	4	1	189983	11.735	0.156
96	15	5	0	222998	1.047	0.750
96	15	5	1	195724	3.828	0.375
101	22	1	0	280250	5.875	0.438
101	22	1	1	249363	9.453	1.063
100	22	2	0	304105	1.984	0.500
100	22	2	1	275585	4.844	2.313
97	22	3	0	273673	2.000	1.234
97	22	3	1	246640	4.531	0.453
109	22	4	0	312437	6.453	4.469
109	22	4	1	280172	12.594	0.594
106	22	5	0	288365	6.219	2.172
106	22	5	1	263434	10.437	1.937
183	15	1	0	354854	5.516	1.344
183	15	1	1	313473	5.687	0.531
184	15	2	0	357438	6.781	0.344
184	15	2	1	314952	7.203	5.469
188	15	3	0	373921	7.032	4.485
188	15	3	1	324686	6.250	1.218
186	15	4	0	365240	6.843	2.937

Continúa en la siguiente página ...

Tabla B.5 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
186	15	4	1	324335	7.329	3.954
185	15	5	0	369152	6.937	0.328
185	15	5	1	322771	6.375	1.656
185	22	1	0	489407	4.218	1.922
185	22	1	1	442640	3.469	1.516
192	22	2	0	548588	4.063	1.625
192	22	2	1	503761	3.234	2.953
184	22	3	0	482731	4.969	0.703
184	22	3	1	440671	4.328	1.922
183	22	4	0	485025	6.641	0.922
183	22	4	1	440740	5.625	3.421
185	22	5	0	481267	4.468	4.468
185	22	5	1	441552	3.438	3.297

Tabla B.6: Resultados GRASP con búsqueda local super rápida

n	$Q_{\text{máx}}$	Núm. Inst.	F	costo	Tiempo Total	Tiempo mejor sol.
42	15	1	0	111414	0.156	0.031
42	15	1	1	90934	1.031	1.000
42	15	1	2	88874	1.500	0.672
42	15	1	3	82973	2.688	0.266
42	15	1	4	80473	4.468	0.843
42	15	2	0	112218	0.313	0.079
42	15	2	1	93439	1.937	1.609

Continúa en la siguiente página ...

Tabla B.6 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
42	15	2	2	86529	4.266	0.657
42	15	2	3	82344	9.094	3.016
42	15	2	4	80011	11.266	0.157
42	15	3	0	111547	0.219	0.047
42	15	3	1	90763	1.234	0.047
42	15	3	2	84838	2.156	0.094
42	15	3	3	81690	3.360	0.797
42	15	3	4	78942	3.594	2.250
42	15	4	1	98202	0.078	0.047
42	15	4	2	90539	0.391	0.110
42	15	4	3	90577	0.453	0.141
42	15	4	4	86031	1.188	0.125
42	15	5	0	112494	0.078	0.046
42	15	5	1	92730	1.109	0.031
42	15	5	2	84291	3.047	0.359
42	15	5	3	80146	3.609	1.578
42	15	5	4	77221	4.547	0.172
42	22	1	0	129292	0.766	0.110
42	22	1	1	107855	1.453	0.750
42	22	1	2	104007	2.344	0.672
42	22	1	3	100300	2.937	0.312
42	22	1	4	99929	3.188	0.063
42	22	2	0	130328	0.765	0.062
42	22	2	1	116876	2.625	0.016
42	22	2	2	108268	3.859	0.141
42	22	2	3	103889	5.219	0.407

Continúa en la siguiente página ...

Tabla B.6 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
42	22	2	4	102822	5.547	0.125
42	22	3	0	133664	0.234	0.047
42	22	3	1	117340	1.422	0.125
42	22	3	2	110464	2.906	1.016
42	22	3	3	107117	3.188	1.157
42	22	3	4	107117	3.953	1.312
42	22	4	0	133609	0.109	0.031
42	22	4	1	122655	0.657	0.188
42	22	4	2	122103	0.672	0.203
42	22	4	3	116109	1.390	0.109
42	22	4	4	116202	1.641	0.219
42	22	5	0	135173	0.844	0.047
42	22	5	1	111888	2.328	0.671
42	22	5	2	108869	3.047	0.125
42	22	5	3	106522	4.656	1.453
42	22	5	4	102978	7.594	0.047
102	15	1	0	226289	10.875	4.859
102	15	1	1	185606	57.891	3.375
104	15	2	0	213084	31.000	2.609
104	15	2	1	179176	73.796	26.265
104	15	3	0	228584	13.579	3.110
104	15	3	1	195522	34.468	30.921
105	15	4	0	222838	26.578	3.703
105	15	4	1	189931	66.766	0.797
96	15	5	0	222998	4.203	2.969
96	15	5	1	195544	21.000	1.828

Continúa en la siguiente página ...

Tabla B.6 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
101	22	1	0	280250	25.485	1.281
101	22	1	1	248824	47.250	5.078
100	22	2	0	304105	7.562	1.390
100	22	2	1	275585	22.953	10.688
97	22	3	0	273673	7.797	2.297
97	22	3	1	246640	21.609	1.547
109	22	4	0	312437	26.235	17.719
109	22	4	1	280172	58.672	2.265
106	22	5	0	288365	28.953	9.422
106	22	5	1	263434	51.547	8.953
183	15	1	0	354764	26.188	18.922
183	15	1	1	312384	31.312	3.062
184	15	2	0	356596	30.781	16.047
184	15	2	1	313319	40.454	33.922
188	15	3	0	373753	29.125	5.687
188	15	3	1	324686	32.531	6.125
186	15	4	0	364368	29.391	29.375
186	15	4	1	324335	38.391	20.313
185	15	5	0	369444	29.968	18.625
185	15	5	1	322077	36.266	22.735
185	22	1	0	489407	16.922	7.313
185	22	1	1	442640	15.781	7.000
192	22	2	0	547884	15.531	6.641
192	22	2	1	503761	14.579	13.157
184	22	3	0	483635	25.906	3.953
184	22	3	1	440671	23.328	10.203

Continúa en la siguiente página ...

Tabla B.6 ... Continuación de la página anterior

n	$Q_{\text{máx}}$	Núm. Inst.	g	costo	Tiempo Total	Tiempo mejor sol.
183	22	4	0	485025	25.141	3.500
183	22	4	1	440740	27.171	16.515
185	22	5	0	481267	18.813	18.813
185	22	5	1	441552	16.297	15.562

Tabla B.7: Resultados GRASP con diferentes valores para α y $\beta = 0$

				$\alpha = 0, \quad \beta = 0$			$\alpha = 0.5, \quad \beta = 0$		
n	$Q_{\text{máx}}$	Núm. Ins.	F	costo	Tiempo Total	Tiempo mejor s.	costo	Tiempo Total	Tiempo mejor s.
42	15	1	0	111414	0.000	0.000	111256	0.375	0.047
42	15	1	1	97749	0.000	0.000	93220	1.093	0.140
42	15	1	2	92565	0.015	0.015	88874	1.469	0.125
42	15	1	3	87247	0.015	0.015	82996	1.735	1.000
42	15	1	4	83027	0.031	0.031	83027	3.109	0.047
42	15	2	0	113875	0.000	0.000	112218	0.563	0.016
42	15	2	1	93997	0.015	0.015	93439	1.297	0.829
42	15	2	2	87675	0.031	0.031	87675	1.985	0.016
42	15	2	3	84092	0.031	0.031	84092	2.937	0.031
42	15	2	4	82041	0.031	0.031	82041	3.172	0.031
42	15	3	0	111547	0.000	0.000	111547	0.281	0.016
42	15	3	1	92306	0.015	0.015	90763	0.907	0.063
42	15	3	2	89393	0.016	0.016	85707	1.39	0.250
42	15	3	3	85838	0.015	0.015	81365	2.266	0.157
42	15	3	4	79120	0.032	0.032	79120	2.703	0.031
42	15	4	1	98646	0.000	0.000	98646	0.188	0.016
42	15	4	2	95711	0.000	0.000	93548	0.75	0.016

Continúa en la siguiente página ...

Tabla B.7 ... Continuación de la página anterior

				$\alpha = 0, \quad \beta = 0$			$\alpha = 0.5, \quad \beta = 0$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.
42	15	4	3	95007	0.016	0.016	91187	0.844	0.016
42	15	4	4	89410	0.015	0.015	87677	1.735	0.797
42	15	5	0	112129	0.000	0.000	111739	0.156	0.156
42	15	5	1	92793	0.015	0.015	91641	1.031	0.641
42	15	5	2	86466	0.016	0.016	84352	1.406	0.609
42	15	5	3	80647	0.015	0.015	80647	1.86	0.016
42	15	5	4	79906	0.015	0.015	79906	2.219	0.031
42	22	1	0	130006	0.000	0.000	129292	0.656	0.015
42	22	1	1	110011	0.016	0.016	110011	1.218	0.015
42	22	1	2	106871	0.016	0.016	106871	1.375	0.016
42	22	1	3	102309	0.015	0.015	102309	1.563	0.032
42	22	1	4	101048	0.016	0.016	101048	1.859	0.016
42	22	2	0	130328	0.016	0.016	130328	0.656	0.015
42	22	2	1	119849	0.016	0.016	117811	1.093	0.656
42	22	2	2	114040	0.016	0.016	108987	1.281	0.187
42	22	2	3	107980	0.016	0.016	107242	1.219	0.031
42	22	2	4	106963	0.015	0.015	106225	1.25	0.016
42	22	3	0	133664	0.015	0.015	133664	0.594	0.000
42	22	3	1	122397	0.016	0.016	118399	0.797	0.500
42	22	3	2	113482	0.016	0.016	111659	1.453	0.172
42	22	3	3	107117	0.015	0.015	107117	1.344	0.016
42	22	3	4	107117	0.015	0.015	107117	1.61	0.032
42	22	4	0	138067	0.016	0.016	133609	0.312	0.000
42	22	4	1	123661	0.015	0.015	122314	0.766	0.016
42	22	4	2	124673	0.000	0.000	120937	0.829	0.750
42	22	4	3	116010	0.015	0.015	115540	1.297	0.016
42	22	4	4	116672	0.016	0.016	116202	1.594	0.016

Continúa en la siguiente página ...

Tabla B.7 ... Continuación de la página anterior

				$\alpha = 0, \quad \beta = 0$			$\alpha = 0.5, \quad \beta = 0$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.
42	22	5	0	135711	0.000	0.000	135173	0.734	0.016
42	22	5	1	119450	0.016	0.016	112936	0.797	0.500
42	22	5	2	108699	0.015	0.015	108699	1.141	0.047
42	22	5	3	108097	0.015	0.015	107324	1.532	0.532
42	22	5	4	103896	0.016	0.016	103896	2.25	0.016
102	15	1	0	228844	0.062	0.062	227567	4.641	3.250
102	15	1	1	188221	0.078	0.078	185925	8.063	5.688
104	15	2	0	218144	0.078	0.078	214107	6.359	0.063
104	15	2	1	188292	0.078	0.078	179397	9.078	5.359
104	15	3	0	229708	0.047	0.047	228557	4.781	4.125
104	15	3	1	206813	0.078	0.078	198345	7.172	2.265
105	15	4	0	229984	0.063	0.063	223585	6.312	0.078
105	15	4	1	193886	0.078	0.078	188912	9.594	2.703
96	15	5	0	230642	0.031	0.031	223175	2.579	0.407
96	15	5	1	205113	0.047	0.047	195058	5.89	2.578
101	22	1	0	281907	0.062	0.062	280563	5.531	4.875
101	22	1	1	258680	0.063	0.063	251327	6.797	4.500
100	22	2	0	310794	0.031	0.031	303879	3.656	0.172
100	22	2	1	283760	0.062	0.062	276121	5.282	3.610
97	22	3	0	280869	0.047	0.047	273301	3.969	2.297
97	22	3	1	251828	0.047	0.047	248159	5.312	4.078
109	22	4	0	315469	0.063	0.063	313491	6.406	3.578
109	22	4	1	289733	0.079	0.079	283032	8.109	0.250
106	22	5	0	292081	0.063	0.063	289390	5.609	5.500
106	22	5	1	275347	0.062	0.062	264681	6.797	3.188
183	15	1	0	363566	0.218	0.218	354692	26.485	23.328
183	15	1	1	318354	0.375	0.375	308482	39.547	34.891

Continúa en la siguiente página ...

Tabla B.7 ... Continuación de la página anterior

				$\alpha = 0, \quad \beta = 0$			$\alpha = 0.5, \quad \beta = 0$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.
184	15	2	0	362881	0.328	0.328	353962	42.625	18.406
184	15	2	1	317164	0.437	0.421	314057	48.578	10.188
188	15	3	0	375011	0.344	0.344	372097	38.484	0.922
188	15	3	1	330584	0.313	0.313	323578	31.343	15.281
186	15	4	0	366896	0.329	0.329	360833	32.875	22.796
186	15	4	1	330217	0.422	0.422	324860	49.515	38.375
185	15	5	0	376698	0.281	0.281	367092	28.297	22.313
185	15	5	1	334708	0.250	0.250	323908	27.187	16.281
185	22	1	0	494328	0.250	0.250	489272	25.172	19.765
185	22	1	1	449292	0.219	0.219	436001	22.094	15.422
192	22	2	1	512127	0.219	0.219	500992	20.266	6.297
184	22	3	0	491644	0.265	0.265	480498	26.36	6.703
184	22	3	1	455390	0.266	0.266	439077	26.281	5.219
183	22	4	0	494311	0.297	0.297	481090	29.406	17.359
183	22	4	1	444809	0.312	0.312	435555	27.266	15.954
185	22	5	0	484214	0.250	0.250	480228	24.016	3.750
185	22	5	1	451106	0.250	0.250	440505	19.516	11.359

Tabla B.8: Resultados GRASP con diferentes valores para α y $\beta = 1$

				$\alpha = 0, \quad \beta = 1$			$\alpha = 0.5, \quad \beta = 1$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.
42	15	1	0	111256	0.047	0.015	111256	0.672	0.031
42	15	1	1	91380	0.125	0.094	90774	2.063	1.063
42	15	1	2	88874	2.360	0.078	88874	2.250	0.890

Continúa en la siguiente página ...

Tabla B.8 ... Continuación de la página anterior

				$\alpha = 0, \quad \beta = 1$			$\alpha = 0.5, \quad \beta = 1$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.
42	15	1	3	82407	3.125	0.171	82533	3.156	2.578
42	15	1	4	80360	4.235	1.047	80363	4.344	1.203
42	15	2	0	113136	0.031	0.015	112218	0.172	0.047
42	15	2	1	90900	2.172	0.453	90900	2.250	0.672
42	15	2	2	86077	3.312	0.984	86077	3.360	0.032
42	15	2	3	82344	4.547	0.125	82344	4.391	1.141
42	15	2	4	80011	5.484	1.797	80011	5.594	2.922
42	15	3	0	111547	0.032	0.016	111547	0.328	0.015
42	15	3	1	90763	1.843	0.328	90763	1.844	0.547
42	15	3	2	84838	2.469	2.282	84838	2.438	1.750
42	15	3	3	80940	3.312	1.016	80778	3.375	3.313
42	15	3	4	78637	4.406	0.453	78683	4.391	0.484
42	15	4	1	98646	0.031	0.015	98646	0.453	0.016
42	15	4	2	94686	0.109	0.031	92535	2.047	0.891
42	15	4	3	89041	2.343	1.015	89528	2.360	1.391
42	15	4	4	86020	3.062	0.359	86020	3.157	0.032
42	15	5	0	112129	0.031	0.015	111951	0.063	0.016
42	15	5	1	91180	0.313	0.047	90746	1.828	0.562
42	15	5	2	84292	2.891	2.234	84291	2.906	1.750
42	15	5	3	79853	3.547	2.125	79853	3.672	2.359
42	15	5	4	77221	4.500	0.422	77221	4.422	0.094
42	22	1	0	129292	0.078	0.016	129292	0.219	0.016
42	22	1	1	107741	1.625	0.110	107855	1.578	0.469
42	22	1	2	104007	2.156	1.734	104007	2.188	0.563
42	22	1	3	100236	2.547	0.844	100236	2.515	1.390
42	22	1	4	100300	2.641	1.110	99929	2.750	1.031
42	22	2	0	130328	0.109	0.016	130328	0.969	0.016

Continúa en la siguiente página ...

Tabla B.8 ... Continuación de la página anterior

				$\alpha = 0, \quad \beta = 1$			$\alpha = 0.5, \quad \beta = 1$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.
42	22	2	1	117085	1.735	1.407	117300	1.734	0.937
42	22	2	2	108268	2.375	0.438	108268	2.328	0.094
42	22	2	3	103856	2.937	0.218	103889	2.969	1.188
42	22	2	4	102156	3.157	1.422	102156	3.156	0.312
42	22	3	0	133664	0.063	0.000	133664	0.234	0.000
42	22	3	1	116916	1.453	0.281	116916	1.437	0.172
42	22	3	2	110464	1.937	0.609	110464	1.954	0.594
42	22	3	3	107117	2.250	0.016	107117	2.297	0.016
42	22	3	4	106934	2.515	0.515	107117	2.532	0.016
42	22	4	0	138067	0.032	0.000	133528	0.328	0.062
42	22	4	1	122655	1.266	0.016	122314	1.218	0.047
42	22	4	2	120317	1.546	0.312	120317	1.485	0.266
42	22	4	3	113170	2.063	1.250	113170	2.015	0.375
42	22	4	4	112634	2.344	1.969	113073	2.296	0.359
42	22	5	0	135173	0.141	0.016	135173	0.953	0.016
42	22	5	1	111888	1.593	0.359	111888	1.610	1.454
42	22	5	2	108699	1.938	0.000	108293	1.906	1.890
42	22	5	3	106185	2.625	1.562	106316	2.578	0.640
42	22	5	4	102978	3.031	0.938	102978	3.047	1.938
102	15	1	0	228241	0.750	0.359	226355	8.125	1.047
102	15	1	1	187969	16.093	5.953	188079	16.125	9.969
104	15	2	0	213084	11.735	3.563	213084	11.750	7.953
104	15	2	1	182045	17.828	2.281	182312	18.031	18.031
104	15	3	0	229708	0.719	0.047	228094	10.188	0.063
104	15	3	1	197839	15.625	11.562	197921	15.703	1.500
105	15	4	0	222403	10.688	5.641	222934	11.047	4.812
105	15	4	1	192219	17.015	0.859	191841	16.985	4.360

Continúa en la siguiente página ...

Tabla B.8 ... Continuación de la página anterior

				$\alpha = 0, \quad \beta = 1$			$\alpha = 0.5, \quad \beta = 1$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.
96	15	5	0	223253	0.328	0.156	223224	3.781	0.578
96	15	5	1	197227	11.625	1.188	196964	11.875	6.360
101	22	1	0	281816	2.532	1.094	279061	8.468	2.875
101	22	1	1	249642	11.656	7.469	249271	11.469	6.172
100	22	2	0	303931	1.594	0.891	303613	3.875	0.453
100	22	2	1	276026	9.765	0.172	275699	9.672	0.157
97	22	3	0	274610	1.063	0.891	273301	5.453	4.328
97	22	3	1	246143	9.422	0.344	246158	9.344	0.687
109	22	4	0	312296	5.750	2.609	312296	9.859	3.750
109	22	4	1	282217	13.187	2.172	280168	13.047	9.672
106	22	5	0	287435	9.516	2.594	287435	9.422	2.093
106	22	5	1	262640	12.219	7.391	262704	12.500	3.140
183	15	1	0	356298	44.125	20.562	354554	60.843	1.828
183	15	1	1	307774	66.688	11.157	311296	74.469	28.844
184	15	2	0	355513	71.593	38.406	355338	70.454	13.563
184	15	2	1	311535	89.485	33.172	314459	89.125	85.046
188	15	3	0	369391	51.594	0.594	369974	51.453	33.937
188	15	3	1	322442	64.610	8.313	326148	64.703	34.031
186	15	4	0	361452	50.640	42.156	362620	59.938	22.750
186	15	4	1	323160	66.391	47.828	321503	60.047	46.250
185	15	5	0	368104	21.422	3.453	369021	45.016	34.188
185	15	5	1	322361	56.172	49.938	321006	56.500	19.313
185	22	1	0	487202	35.421	10.671	485956	35.594	17.047
185	22	1	1	434490	40.109	21.015	433924	40.953	8.453
192	22	2	1	493892	39.562	14.797	493635	39.953	13.282
184	22	3	0	480527	38.250	3.390	481791	38.718	16.843
184	22	3	1	433863	42.406	39.578	435010	42.625	25.094

Continúa en la siguiente página ...

Tabla B.8 ... Continuación de la página anterior

				$\alpha = 0, \quad \beta = 1$			$\alpha = 0.5, \quad \beta = 1$		
		Núm.			Tiempo	Tiempo		Tiempo	Tiempo
n	$Q_{\text{máx}}$	Ins.	F	costo	Total	mejor s.	costo	Total	mejor s.
183	22	4	0	481314	37.938	3.453	481330	37.656	31.968
183	22	4	1	432039	44.219	3.891	431649	44.234	20.219
185	22	5	0	479177	35.797	3.375	478617	35.750	9.000
185	22	5	1	432812	40.734	18.328	434816	40.609	3.109

Tabla B.9: Resultados VNS para diferentes valores de r

n	Inst.	$r=5$	$r=6$	$r=7$	$r=8$	$r=9$	$r=10$	$r=11$	$r=12$	$r=13$	$r=15$	$r=20$
41	1	6425.17	6425.17	6425.17	6425.17	6425.17	6425.17	6425.17	6425.17	6425.17	6425.17	6425.17
	2	6173.31	6173.31	6173.31	6173.31	6173.31	6173.31	6173.31	6173.31	6173.31	6173.31	6173.31
	3	6355.62	6355.62	6355.62	6355.62	6355.62	6355.62	6355.62	6355.62	6355.62	6355.62	6355.62
	4	6902.71	6902.71	6902.71	6902.71	6902.71	6902.71	6902.71	6902.71	6902.71	6902.71	6902.71
	5	6832.63	6935.43	6832.63	6832.63	6832.63	6832.63	6832.63	6832.63	6832.63	6832.63	6832.63
62	1	9338.92	9421.68	9303.63	9344.73	9303.63	9338.92	9316.82	9303.63	9316.82	9298.06	9298.06
	2	8851.1	8851.1	8851.1	8851.1	8851.1	8851.1	8851.1	8851.1	8766.68	8766.68	8765.2
	3	8865.25	8865.25	8865.25	8865.25	8893.57	8893.57	8813.65	8870.67	8813.65	8870.67	8893.57
	4	9089.24	9089.24	9089.24	9089.24	9089.24	9089.24	9089.24	9089.24	9089.24	9089.24	9089.24
	5	8946.49	8942.57	8946.49	8942.57	8942.57	8942.57	8942.57	8942.57	8942.57	8942.57	8942.57
82	1	10931.77	10902.77	10902.77	11008.53	10891.37	10902.77	10788.73	10788.73	10880.42	10891.37	10788.73
	2	10520.13	10431.29	10430.65	10451.85	10471.07	10471.07	10430.65	10430.65	10430.65	10431.29	10430.65
	3	10182.39	10086.87	10086.87	10075.53	10109.25	10109.25	10086.18	10086.18	10075.53	10075.53	10075.53
	4	10555.07	10555.07	10555.07	10555.07	10550.06	10550.06	10550.06	10550.06	10550.06	10550.06	10550.06
	5	10769.95	10769.95	10689.07	10756.35	10724.22	10701.96	10741.22	10662.9	10662.9	10662.21	10662.21
102	1	13302.32	13264.3	13302.32	13189.47	13189.47	13111.88	13148.74	13125.16	13279.87	13111.88	13111.88
	2	13884.61	13875.47	13929.22	13925.32	13883.72	13799.18	13877.77	13799.18	13799.18	13799.18	13799.18
	3	11982.54	11938.14	11945.49	11963	11963	11995.39	11963	11905.12	11902.96	11963	11913.58
	4	12373.72	12373.72	12373.72	12373.72	12373.72	12406.5	12406.5	12373.72	12373.72	12373.72	12303.52
	5	12675.54	12733.46	12595.05	12626.1	12715.38	12585.67	12675.54	12701.56	12575.35	12592.48	12580.2

Apéndice C

Resultados computacionales modelo biobjetivo

Tabla C.1: Número de puntos no dominados obtenidos al finalizar cada fase del MO-BEV

		Número de puntos no dominados en cada fase y tiempo					
	Núm.	Primera fase		Segunda fase		Tercera fase	
n	Inst.	No.puntos ND	tiempo	No.puntos ND	tiempo	No.puntos ND	tiempo
42	1	9	134	46	105	46	0.063
	2	14	137	33	133	35	0.047
	3	10	138	25	90	27	0.032
	4	12	130	62	114	67	0.094
	5	9	108	32	97	46	0.062
62	1	7	368	40	420	56	0.156
	2	8	391	48	642	58	0.234
	3	15	353	31	210	54	0.297
	4	21	323	45	151	73	0.297
	5	12	542	41	575	55	0.344
82	1	15	1029	35	390	57	0.844
	2	15	978	53	691	101	1.093
	3	10	899	47	676	98	1.281
	4	9	904	48	674	68	0.86
	5	6	931	48	683	122	1.718
102	1	17	1643	36	1556	97	2.344
	2	12	1474	58	1339	119	3.609
	3	10	1856	35	1303	151	3.485
	4	16	1811	39	1585	129	2.734
	5	12	1872	55	1496	112	2.906

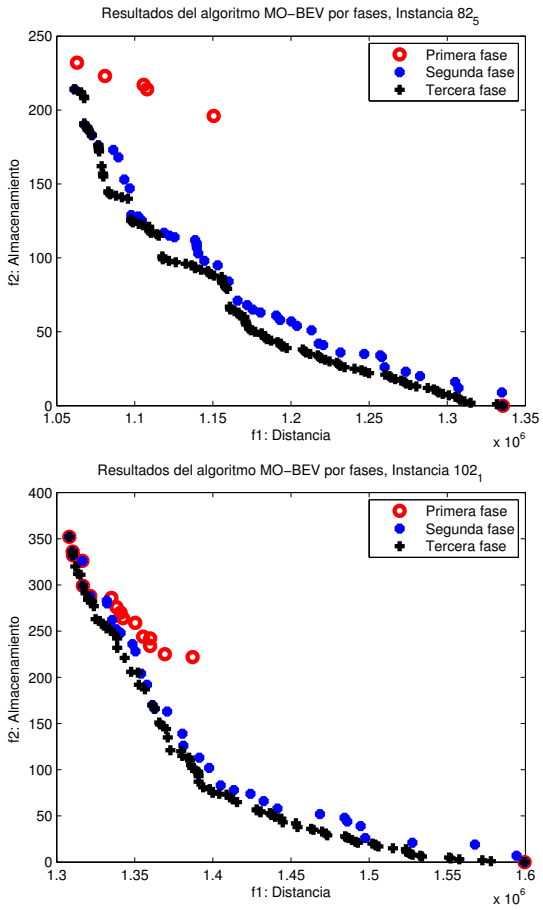


Figura C.1: Resultados por fases del algoritmo MO-BEV para las instancias Instancia82_5 e Instancia102_1

Tabla C.2: Número de puntos no dominados obtenidos con MO-BEV y con NSGA-RF

		Número de puntos no dominados					
	Núm.	(20,5)		(50,10)		(100,10)	
n	Inst.	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF
42	1	46	47	49	47	49	47
	2	33	36	36	34	36	35
	3	27	23	28	31	28	31
	4	67	56	65	68	68	58
	5	46	46	50	50	51	48
62	1	56	71	69	58	73	59
	2	58	52	52	61	56	54
	3	54	47	61	59	60	59
	4	73	79	78	86	82	82
	5	55	54	56	49	51	54
82	1	57	85	65	83	64	102
	2	101	93	108	102	110	105
	3	98	78	101	56	107	94
	4	68	86	80	121	83	112
	5	122	86	129	127	131	125
102	1	97	50	102	116	96	71
	2	119	124	147	173	121	129
	3	151	138	148	151	156	157
	4	129	98	137	99	122	161
	5	112	66	122	81	106	118

Tabla C.3: Extremo derecho del frente de Pareto obtenidos con MO-BEV y con NSGA-RF

		Extremo derecho del frente de Pareto					
	Núm.	(20,5)		(50,10)		(100,10)	
n	Inst.	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF
42	1	(812824 , 0)	(812824 , 0)	(812824 , 0)	(812824 , 0)	(812824 , 0)	(812824 , 0)
	2	(793506 , 0)	(793506 , 0)	(793506 , 0)	(793506 , 0)	(793506 , 0)	(793506 , 0)
	3	(818979 , 0)	(818979 , 0)	(818979 , 0)	(818979 , 0)	(818979 , 0)	(818979 , 0)
	4	(878414 , 0)	(878414 , 0)	(878414 , 0)	(878414 , 0)	(878414 , 0)	(878414 , 0)
	5	(835442 , 0)	(835442 , 0)	(835442 , 0)	(835442 , 0)	(835442 , 0)	(835442 , 0)
62	1	(1158995 , 0)	(1158995 , 0)	(1158995 , 0)	(1158995 , 0)	(1158995 , 0)	(1158995 , 0)
	2	(1053749 , 0)	(1053749 , 0)	(1053749 , 0)	(1058453 , 0)	(1053749 , 0)	(1058453 , 0)
	3	(1085421 , 0)	(1085421 , 0)	(1085421 , 0)	(1091159 , 0)	(1085421 , 0)	(1087058 , 0)
	4	(1121416 , 0)	(1121416 , 0)	(1121416 , 0)	(1121416 , 0)	(1121416 , 0)	(1121416 , 0)
	5	(1132300 , 0)	(1141077 , 0)	(1132300 , 0)	(1132300 , 0)	(1132300 , 0)	(1132847 , 0)
82	1	(1297969 , 0)	(1310686 , 0)	(1297969 , 0)	(1308209 , 0)	(1297969 , 0)	(1299045 , 0)
	2	(1268273 , 0)	(1269733 , 0)	(1268273 , 0)	(1268273 , 0)	(1268273 , 0)	(1282741 , 0)
	3	(1272501 , 0)	(1290542 , 0)	(1272501 , 0)	(1276410 , 0)	(1272501 , 0)	(1272501 , 0)
	4	(1355731 , 0)	(1366642 , 0)	(1355731 , 0)	(1361508 , 0)	(1355731 , 0)	(1361570 , 0)
	5	(1335629 , 0)	(1338282 , 0)	(1335629 , 0)	(1340273 , 0)	(1335629 , 0)	(1343810 , 0)
102	1	(1599502 , 0)	(1607471 , 0)	(1599502 , 0)	(1606972 , 0)	(1599502 , 0)	(1608896 , 0)
	2	(1699606 , 0)	(1710332 , 0)	(1699606 , 0)	(1712380 , 0)	(1699606 , 0)	(1736805 , 0)
	3	(1481641 , 0)	(1496111 , 0)	(1481641 , 0)	(1488728 , 0)	(1481641 , 0)	(1483208 , 0)
	4	(1509677 , 0)	(1522762 , 0)	(1509677 , 0)	(1512131 , 0)	(1509677 , 0)	(1510462 , 0)
	5	(1516542 , 0)	(1525065 , 0)	(1516542 , 0)	(1520486 , 0)	(1516542 , 0)	(1528862 , 0)

Tabla C.4: Extremo izquierdo del frente de Pareto obtenidos con MO-BEV y con NSGA-RF

		Extremo derecho del frente de Pareto					
	Núm.	(20,5)		(50,10)		(100,10)	
n	Inst.	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF	MO-BEV	NSGA-RF
42	1	(642495 , 94)	(642495 , 94)	(642495 , 94)	(643006 , 74)	(642495 , 94)	(643006 , 74)
	2	(617309 , 115)	(617344 , 116)	(617309 , 115)	(617309 , 115)	(617309 , 115)	(617309 , 115)
	3	(635539 , 75)	(666938 , 54)	(635539 , 75)	(635539 , 75)	(635539 , 75)	(635539 , 75)
	4	(690249 , 142)	(697973 , 139)	(690249 , 142)	(690249 , 142)	(690249 , 142)	(698269 , 107)
	5	(683240 , 119)	(698153 , 78)	(683240 , 119)	(691482 , 99)	(683240 , 119)	(683240 , 119)
62	1	(929724 , 186)	(950569 , 168)	(929724 , 186)	(935610 , 192)	(929724 , 186)	(935610 , 192)
	2	(879879 , 153)	(912780 , 138)	(879879 , 153)	(886407 , 142)	(879879 , 153)	(886728 , 142)
	3	(881334 , 196)	(897106 , 134)	(881334 , 196)	(890328 , 146)	(881334 , 196)	(887044 , 149)
	4	(908894 , 200)	(929004 , 156)	(908894 , 200)	(926884 , 173)	(908894 , 200)	(929004 , 156)
	5	(883769 , 127)	(911170 , 126)	(883769 , 127)	(899791 , 67)	(883769 , 127)	(883769 , 127)
82	1	(1071913 , 247)	(1082734 , 183)	(1070018 , 215)	(1100533 , 172)	(1068837 , 215)	(1098138 , 196)
	2	(1039683 , 243)	(1085632 , 177)	(1039683 , 243)	(1073948 , 177)	(1039683 , 243)	(1063023 , 162)
	3	(1008647 , 281)	(1057034 , 176)	(1006237 , 264)	(1047303 , 148)	(1006237 , 264)	(1019849 , 236)
	4	(1054968 , 200)	(1071796 , 177)	(1054968 , 200)	(1081460 , 187)	(1054968 , 200)	(1088771 , 203)
	5	(1061493 , 214)	(1078613 , 168)	(1063059 , 232)	(1067779 , 231)	(1056938 , 236)	(1093529 , 231)
102	1	(1308127 , 352)	(1377409 , 171)	(1308127 , 352)	(1368557 , 204)	(1308127 , 352)	(1373184 , 229)
	2	(1382731 , 371)	(1431446 , 237)	(1382731 , 371)	(1420575 , 267)	(1382731 , 371)	(1430024 , 245)
	3	(1189803 , 259)	(1233625 , 181)	(1189803 , 259)	(1211432 , 213)	(1189803 , 259)	(1207425 , 226)
	4	(1239815 , 256)	(1290587 , 157)	(1237214 , 256)	(1271518 , 173)	(1239815 , 256)	(1242507 , 205)
	5	(1256006 , 290)	(1282943 , 223)	(1256006 , 290)	(1274773 , 259)	(1256006 , 290)	(1274248 , 229)

Tabla C.5: Tiempo utilizado por MO-BEV y NSGA-RF con los diferentes parámetros

		Tiempo		
n	Núm.Inst.	(20,5)	(50,10)	(100,10)
42	1	239	794	1590
	2	270	667	645
	3	228	492	672
	4	244	782	695
	5	205	607	1089
62	1	788	3931	9539
	2	1033	4140	6516
	3	563	2583	2587
	4	474	3182	3730
	5	1116	3854	7611
82	1	1419	4857	12775
	2	1670	6866	16102
	3	1574	5435	11628
	4	1573	5416	9490
	5	1611	3195	9571
102	1	3201	13161	18777
	2	2813	9433	6760
	3	3162	13888	28908
	4	3397	7912	20536
	5	3372	11648	17109

Tabla C.6: Comparación de resultados C(MO-BEV,NSGA-RF) y C(NSGA-RF,MO-BEV

n		Resultados MO-BEV			Resultados NSGA-RF		
n	Núm. Inst.	Núm. puntos frente Pareto	Núm. puntos domina–dos por NSGA-RF	C(NSGA-RF, MO-BEV)	Núm. puntos frente Pareto	Núm. puntos domina–dos por MO-BEV	C(MO-BEV, NSGA-RF)
42	1	49	0	0	47	1	0.02
	2	36	0	0	34	0	0
	3	28	0	0	31	2	0.06
	4	65	0	0	68	18	0.26
	5	50	0	0	50	0	0
62	1	69	20	0.29	58	19	0.33
	2	52	2	0.04	61	19	0.31
	3	61	10	0.16	59	16	0.27
	4	78	4	0.05	86	18	0.21
	5	56	9	0.16	49	8	0.16
82	1	65	21	0.32	83	28	0.34
	2	108	26	0.24	102	54	0.53
	3	101	7	0.07	56	29	0.52
	4	80	9	0.11	121	88	0.73
	5	129	41	0.32	127	70	0.55
102	1	102	32	0.31	116	44	0.38
	2	147	90	0.61	173	45	0.26
	3	148	86	0.58	151	46	0.30
	4	137	74	0.54	99	35	0.35
	5	122	59	0.48	81	23	0.28
Promedio				**0.21**			**0.29**

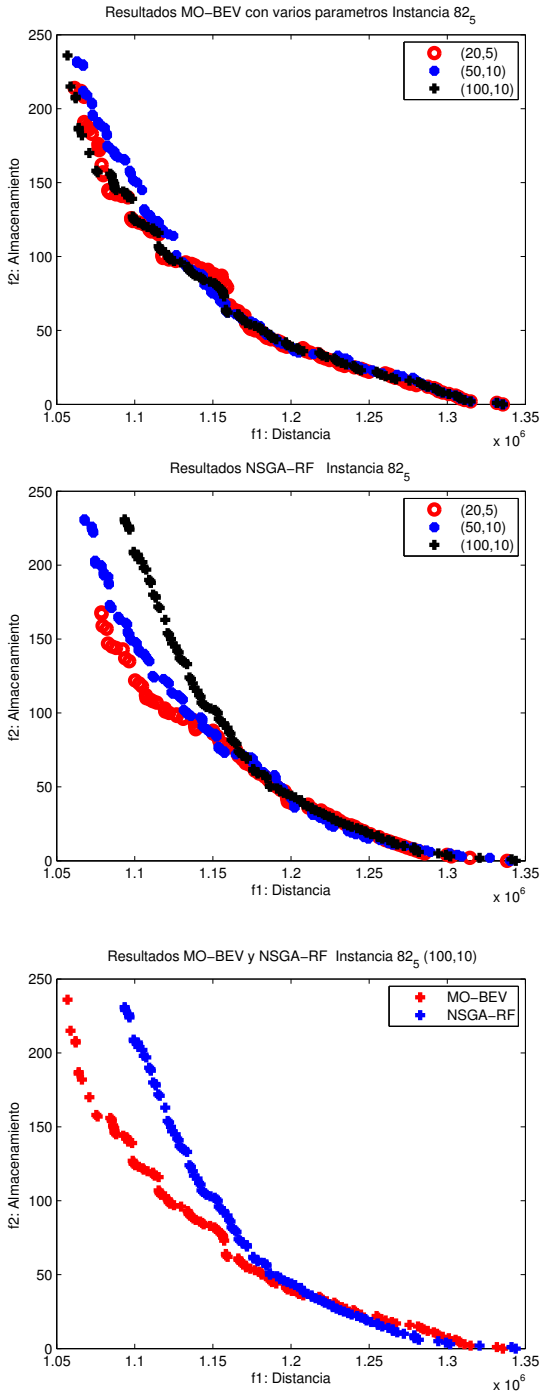


Figura C.2: Frentes de Pareto obtenidos con MO-BEV y NSGA-RF en una instancia de 82 órdenes

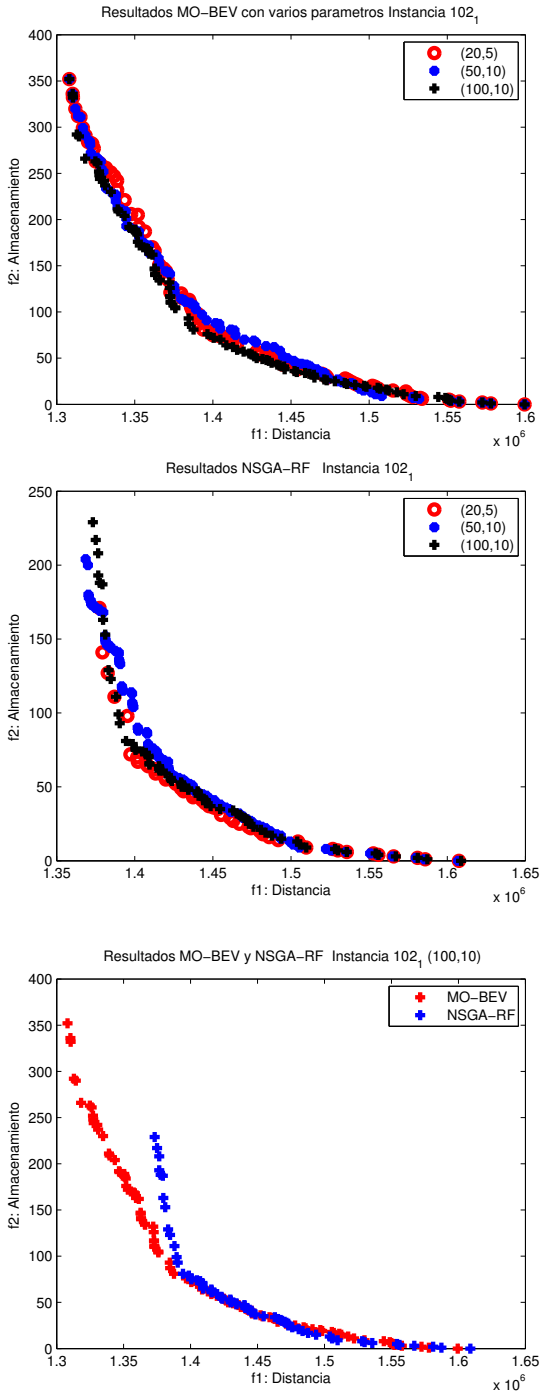


Figura C.3: Frentes de Pareto obtenidos con MO-BEV y NSGA-RF en una instancia de 102 órdenes

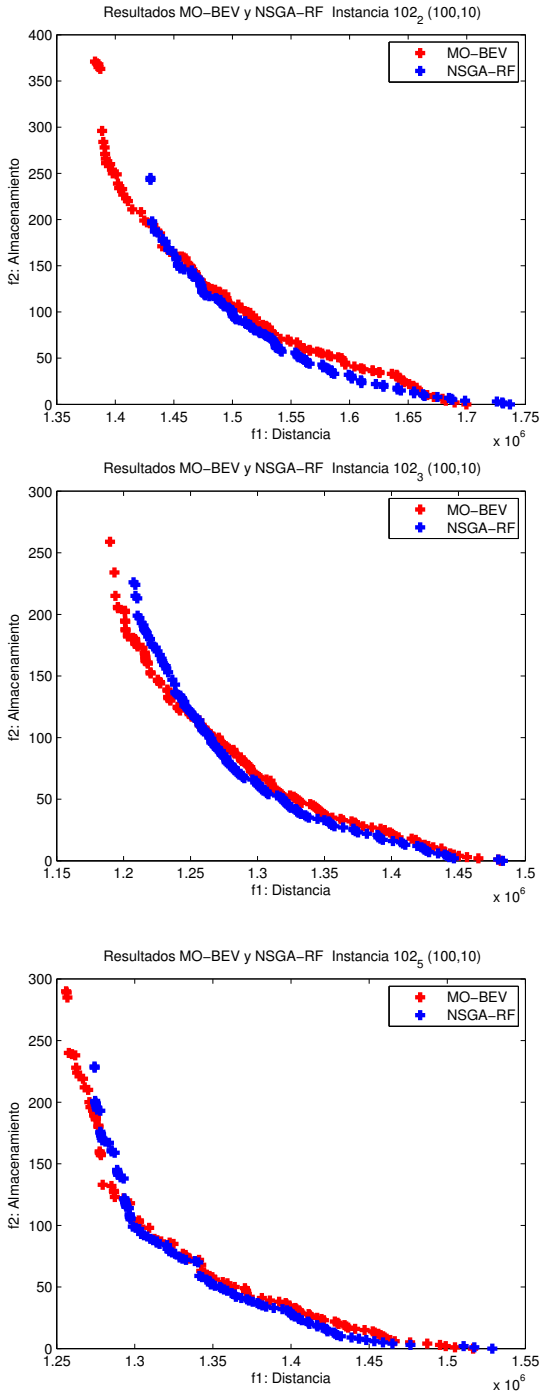


Figura C.4: Comparación de los frentes de Pareto obtenidos con MO-BEV y NSGA-RF en varias instancias de 102 órdenes

Tabla C.7: Comparación de resultados de NSGA-RF variando la probabilidad de mutación.

		A		B		A		D	
n	Num. Inst.	Num Pts. Frente Pareto	C(B,A)	Num Pts. Frente Pareto	C(A,B)	Num Pts. Frente Pareto	C(D,A)	Num Pts. Frente Pareto	C(A,D)
42	1	47	0.02	47	0.13	47	0.00	46	0
	2	36	0.14	33	0	36	0.05	31	0
	3	23	0.08	29	0	23	0.08	28	0
	4	56	0.14	66	0.18	56	0.08	66	0.79
	5	46	0.08	50	0.04	46	0.08	52	0.04
62	1	71	0.52	45	0.11	71	0.11	54	0.46
	2	52	0.46	49	0.20	52	0.33	67	0.45
	3	47	0.38	64	0.34	47	0.23	58	0.38
	4	79	0.05	83	0.14	79	0.05	83	0.05
	5	54	0.46	42	0.09	54	0.56	51	0.00
82	1	85	0.15	70	0.41	85	0.19	52	0.44
	2	93	0.43	92	0.20	93	0.65	95	0.19
	3	78	0.54	84	0.21	78	0.63	88	0.18
	4	86	0.56	62	0.19	86	0.23	98	0.66
	5	86	0.50	102	0.43	86	0.05	133	0.69
102	1	50	0.02	61	0.79	50	0.02	92	0.71
	2	124	0.30	154	0.55	124	0.30	118	0.59
	3	138	0.14	133	0.77	138	0.99	125	0.00
	4	98	0.14	140	0.61	98	0.81	88	0.10
	5	66	0.39	72	0.26	66	0.35	117	0.00
Promedio			0.28		0.28		0.29		0.29

A=Resultados con NSGA-RF con probabilidad de mutación 0.1

B=Resultados con NSGA-RF con probabilidad de mutación 0.01

D=Resultados con NSGA-RF con probabilidad de mutación 0.02

C(A,B) = métrica de Ziztler

Bibliografía

[1] B. Adenso–Díaz, M. Laguna, Fine–tuning of algorithms using fractional experimental designs and local search, *Operational Research*, (54) 99–114, 2006.

[2] R. Ahuja, T. Magnanti, J. Orlin, *Network Flows*, Prentice Hall, New Jersey, 1993.

[3] D. Aksen, N. Aras, Customer Selection and Profit Maximization in Vehicle Routing Problems, *Operations Research Proceedings 2005*, Springer, 37–42, 2005.

[4] C. Alabas-Uslu, A self-tuning heuristic for a multi-objective vehicle routing problem, *Journal of the Operational Researcj Society*, (59)7 988-996, 2008.

[5] D. Ambrosino, A. Sciomachen, A food distribution network problem: A case study, *IMA Journal Management Mathematics* 18(1) 33-53, 2007.

[6] E. Angelelli, M. Speranza, The periodic vehicle routing with intermediate facilities, *European Journal of Operational Research, Elsevier* (137) 233–247, 2002.

[7] J. Alegre, M. Laguna, J. Pacheco, Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts, *European Jounal of Operational Research, Elsevier* (179) 736–746, 2007.

[8] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, D. Naddef G. Rinaldi, Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report RR 949-M, *Université Joseph Fourier* Grenoble, 1995.

[9] R. Baldacci, N. Christofides, A. Mingozzi, An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts, *Mathematical Programming: Series A and B* (115), 351-385, 2008.

[10] R. Baldacci, E. Hadjiconstantinou, A. Mingozzi, An exact algorithm for the capacitated vehicle routing problem based on a two–commodity network flow formulation, *Operations research* (52) 723–738, 2004.

[11] S. Baptista, R. Carvalho, E. Zúquete, A period vehicle routing case study, *European Journal of Operational Research, Elsevier* (139) 220–229, 2002.

[12] J. Beasley, Route–first cluster–second methods for vehicle routing, *Omega*, (11) 403–408, 1983.

[13] R. Beausoleil, MOSS Multiobjective Scatter Search Applied to Nonlinear multiple criteria optimization, *European Journal of Operational Research*, (169) 426–449, 2006.

[14] J. Bentley, Fast Algorithms for Geometric Traveling Salesman Problems, *ORSA Journal on Computing*, (4) 387–411, 1992.

[15] R. Bowerman, B. Hall, P. Calami, A multiobjective optimization approach to urban school bus routing: formulation and solution method, *Transportacion Research*, (29) 123–197, 1995.

[16] O. Braysy, A reactive variable neighborhood search for the vehicle routing problems with time windows, *INFORMS Journal on Computing* (15) 347–368, 2003.

[17] R. Caballero, X. Gandibleux, J. Molina, MOAMP- A Generic Multiobjective Metaheuristic using an Adaptive Memory, Technical Report, *University of Valenciennes*, France, 2004.

[18] R. Caballero, J. Molina y V. Rodríguez, MOAMP: Programación Multiobjetivo mediante un procedimiento de Búsqueda Tabú, *Actas del II Congreso Español de Metaheurísticas y Algoritmos Evolutivos y Bioinspirados: MAEB*, Universidad de Oviedo, España, 153-159, 2003.

[19] N. Christofides, J. Beasley, The Period Routing Problem, Networks, (14) 237–256, 1984.

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

[21] A. Corberán, E. Fernández, M. Laguna, R. Martí, Heuristic solution to the problem of routing school buses with multiple objectives, *Journal of the operational research society*, (53)4, 427–435, 2002.

[22] J.F. Cordeau, M. Gendreau, G. Laporte, A tabu serach heuristic for periodic and multi–depot vehicle routing problems, Networks, (30) 105–199, 1997.

[23] J.F. Cordeau, G. Laporte, A. Mercier, A Unified Tabu Search Heuristic for Vehcile Routing Problems with Time Windows, The Journal of the Operational Research Society, (52) 928–936, 2001

[24] J.F. Cordeau, G. Laporte, M. Savelsberg, D. Vigo. "Vehicle Routing"in C. Barnhart and G. Laporte (Editors), Transportation, handbooks in operations research and managament science, Vol. 14, Elsevier, Amsterdam, pp. 367–428, 2007.

[25] G. Cornuéjols, F. Harche, Polyhedral study of the capacitated vehicle routing problems, *Mathematical Programming*, (60) 21–52, 1993.

[26] G. Dantzing, J. Ramser, The truck dispatching problem, *Management Science*, (6) 81–91, 1959.

[27] K. Deb, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on evolutionary computation*, 6(2) 182–197, 2002.

[28] C. Delgado, J. Pacheco, MinMax vehicle routing problems: application to school transport in the province of Burgos, *Lecture Notes in Economics and Mathematical Systems*, (505) 297-318, 2001.

[29] K. Doerner, A. Focke, W. Gutjahr, Multicriteria tour planning for mobile healthcare facilities in a developing country, *European Journal of Operational Research*, (179) 1078-1096, 2007.

[30] R. Eglese, A. Mercer, B. Sohrabi, The Grocery Superstore Vehicle Scheduling Problem, *The Journal of the Operational Research Society* 56(8) 902-911, 2005.

[31] M. Ehrgott, *Multicriteria Optimization, Second Ed.*, Springer Berlin Heidelberg, 2005.

[32] M. Ehrgott, X. Gandibluex, A Survey and annotated bibliography on multiobjective conbinatorial optimization, *OR Spektrum* (22), 425–460, 2000.

[33] Ö. Ergun, J. Orlin, A. Steele–Feldman, Creating very large scale neighborhood out of smaller ones by compounding moves, *Journal of Heuristics* (12) 115–140, 2006.

[34] T. Feo, M. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Operations Research Letter, (8) 67–71, 1989.

[35] D. Feillet, P. Dejax M. Gendreau, Traveling Salesman Problems with Profits *Transportation Science* (39) 188–205 2005.

[36] T. Feo, M. Resende, Greedy Randomized Adaptive Search Procedures, Journal of Global Optimization, (6) 109–133, 1995.

[37] M.L. Fisher, Optimal solution of vehicle routing problems using minimum k–trees. *Operations Research*, (42) 626–642, 1994.

[38] M. Fisher, R. Jaikumar, A Generalized Assignment Heuristics for Vehicle Routing Problem, Networks (11) 109–124, 1981.

[39] P. Francis, K. Smilowitz, M. Tzur, The Period Vehicle Routing Problem with Service Choice, *Transportation Science*, (40) 439–454, 2006.

[40] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, R. Werneck, Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem, *Mathematical Programming* (106) 491-511, 2006.

[41] M. Gendreau, A. Hertz, G. Laporte, New insertion and postoptimization procedures for the traveling salesman problem, *Operations Research*, (40) 1086–1094, 1992.

[42] B. Gillet, L. Miller, A heuristic algorithm for the vehicle–dispatch problem, *Operations research*, (21) 340–349, 1974.

[43] F. Glover, Heuristics for Integer Programming Using Surrogate Constraints, *Decision Sciences* (8) 156–166, 1977.

[44] F. Glover,(1998) A Template for Scatter Search and Path Relinking, In: *Artificial Evolution, Lecture Notes in Computer Science 1363*, J. K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Editors), Springer-Verlag, 13–54.

[45] F. Glover, Tabu search and adaptive memory programming- Advances, applications and challenges. In R. Barr, R. Helgason, J. Kennington, editors. Interfaces in Computer Science and Operations Research, Kluwer, 1–75, 1996.

[46] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston MA, 1997.

[47] F. Glover, M. Laguna, R. Martí, Fundamentals of scatter search and path relinking, *Control and Cybernetics* (39) 653–684, 2000.

[48] L. Gonçalves, S. Martins, L. Ochi, A GRASP with Adaptive Memory for a Period Vehicle Routing Problem, in *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation, CIMCA2005*, M. Mohammadian (Ed.) 721-727, 2005.

[49] P. Hansen, N Mladenović, An introduction to variable neighborhood search, Metaheuristics: Advances and Trends in local Search Paradigms for Optimization, Kluwer Academic Publishers, Boston, 433–458, 1999.

[50] P. Hansen, N. Mladenović, Variable neighborhood search: Principles and applications, *European Journal of Operational Research* (130) 449–467, 2001.

[51] V. Hemmelmays, K. Doerner, R. Hartl, A variable neighborhood searc heuristic for periodic routing problems, *European Journal of Operational Research* (195) 791–802, 2009.

[52] C. Hsu, S. Hung, H. Li, Vehicle routing problem with time-windows for perishable food delivery, *Journal of Food Engineering* 80(2) 465-475, 2007.

[53] N. Jozefowiez, F. Glover. M. Laguna, Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits, *Journal of Mathematical Modelling and Algorithms* (7) 177–195, 2008.

[54] N. Jozefowiez, F. Semet, and E-G. Talbi, From single-objective to multi-objective vehicle routing problems: Motivations, case studies, and methods, in *The vehicle routing problem: Latest advances and new challenges*, B. L. Golden, S. Raghavan, E. Wasil (Eds), Springer, 445-471, 2008.

[55] N. Jozefowiez, F. Semet, E-G. Talbi, Multi–objective vehicle routing problems, European Journal of Operations Research, (189) 293–309, 2008.

[56] N. Jozefowiez, F. Semet, E-G. Talbi, Target Aiming Pareto search and its applications to the vehicle routing problem with route balancing, *Journal of Heuristics*, 13(5), 455-469, 2007.

[57] A. Keith Austin, An Elementary approach to NP–Completeness, *The American Mathematical Monthly*, 6 (90), 398–399, 1983.

[58] C. Keller, M. Goodchild, The multiobjective vending problem: A generalization of traveling salesman problem, *Environment and Planning B: Planning and Design* (15) 447-460, 1988.

[59] I.Y. Kim, O.L. Weck, Adaptive wighted–sum method for bi–objective optimization: Pareto front generation, *Struct Multidisc Optim* , 29, 149–158, 2005.

[60] M. Laguna, R. Martí, *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Publishers, 2003.

[61] G. Laporte, What you should know about the Vehicle Routing Problem, *Wiley Periodicals , Naval Research Logistics* (54) 811-819, 2007

[62] G. Laporte, M. Gendreau, J-Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem, *International Transactions in Operational Research* 7(4-5) 285–300, 2000.

[63] G. Laporte, Y. Nobert, M. Desrochers, Optimal routing under capacity and distance restrictions, *Operations Research*, (33) 1050–1073, 1985.

[64] T-R Lee, J-H Ueng, A study of vehicle routing problem with load balancing, *International Journal of Physical Distribution and Logistics Management*, (29) 646–648, 1998.

[65] J. Lenstra, A. Rinnooy Kan, Complexity of Routing Vehicle and Scheduling Problems, *Networks* (11) 221-228, 1981.

[66] S. Martello, P. Toth, *Knapsack problems algorithms and computer implementatios*, John Wiley & Sons, 1990.

[67] H. Min, A multiobjective vehicle routing problem with soft time windows: the case of a public library distribution system, *Socio-Economic Planning Sciences* 25(3) 179-188, 1991.

[68] R. Mole, S. Jameson, A sequential route–building algorithm employing a generalized savings criterion, *Operational Research Quarterly* (27) 503–511, 1976.

[69] J. Molina, M. Laguna, R. Martí, R. Caballero, SSPMO: A Scatter Tabu Search Procedure for non linear multiobjective optimization, *INFORMS Journal on Computing*, (19) 91–100, 2007.

[70] D. Mester, O. Bräysy, Actived-guided evolution strategies for large–scale capacitated vehicle routing problems, *Computers & Operations Research*, (34) 2964–2975, 2007.

[71] D. Naddef, G. Rinaldi, Branch and cut algorithms for the capactitated VRP. In: Toth and Vigo (Editors), *The Vehicle Routing Problem*, SIAM. Monographs on Discrete Mathematical and Applications. SIAM Philadelphia, 53–82, 2002.

[72] D. Naso, M. Surico, B. Turchiano, Scheduling Production and Distribution of Rapidly Perishable Materials with Hybrid GA's, *Evolutionary Scheduling* 465-483, 2007.

[73] I. Or, *Traveling salesman-type combinatorial problems and their relation to the logistic of blood banking*, PhD Thesis, Departament of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 1976.

[74] I. Osman, J. Kelly (Editors), *Meta–Heuristics: Theory & Applications*, Kluwer, Boston, 1996.

[75] A. Osvald, L. Stirn, A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food, *Journal of Food Engineering* 85(2) 285-295, 2008.

[76] J. Pacheco, R. Martí, Tabu Search for a multi–objective routing problem, *Journal of Operations Research Society*, (57) 29–37, 2006.

[77] Ch. Papadimitriou, *Computational Complexity*, Addison Wesley Longman, 1995.

[78] D. Pisinger, S. Ropke, A general heuristic for vehicle routing problems, *Computers & Operations Research*, (34) 2403–2435, 2007.

[79] P. Prado, L. Lopes, Scatter search for the fleet size and mix vehicle routing problem with time windows, *Central European Journal of Operations Research* (15) 351–368, 2007.

[80] C. Prins, A simple and effective evolutionary algorithm for the vehicle routing problem, *Computers & Operations Research*, (31) 1985–2002, 2004.

[81] C. Romero, *Teoría de la decisión multicriterio: conceptos, técnicas y aplicaciones*, Alianza Universidad Textos. Alianza Editorial, Madrid, 1993.

[82] S. Ropke, J.-F. Cordeau, G. Laporte, Models and a Branch-and-Cut Algorithm for Pickup and Delivery Problems with Time Windows, *Networks* (49) 258-272, 2007.

[83] R. Russell, and W. Chiang, Scatter Search for the Vehicle Routing Problem with Time Windows, *European Journal of Operational Research* (169) 606-622, 2006.

[84] E. Taillard, P. Badeau, M. Gendreau, F. Guertin, J–Y Potvin, A tabu search heuristic for the vehicle routing problem with soft time windows, *Transportation Science* (31) 170–186, 1997.

[85] C. Tarantilis, C. Kiranoudis, A meta-heuristic algorithm for the efficient distribution of perishable foods *Journal of Food Engineering* 50(1) 1-9, 2001.

[86] G. Tirado *El problema de rutas de vehículos (VRP)*. Tesis de Maestría, Universidad Computlense de Madrid. Madrid, España, 2006.

[87] P. Toth, D. Vigo (Editors), *The Vehicle Routign Problem*, SIAM. Monographs on Discrete Mathematical and Applications. SIAM Philadelphia, 2002.

[88] D. Valladares, Una aplicación computacional basada en la heurística GENI para dar solución al problema del Agente Viajero, Tesis de Licenciatura, Facultad de Ciencias Físico Matemáticas, Universidad Autónoma de Nuevo León, México, Julio 2008.

[89] M. Wen, J. Cordeau, G. Laporte, J. Larsen, The dynamic multi-period vehicle routing problem, *Computers and Operations Research*, (37) 1615–1623, 2010.

[90] E. Zitzler, L. Thiele, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation* (3) 257-271, 1999.

Ficha autobiográfica

Irma Delia García Calvillo

Candidato para el grado de Doctor en Ingeniería
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Un enfoque metaheurístico para un problema de ruteo con flexibilidad en las fechas de entrega

Nací en la ciudad de Saltillo, Coahuila. Obtuve el título de Licenciada en Matemáticas Aplicadas por la Universidad Autónoma de Coahuila. Realicé los estudios de Maestría en Ciencias Matemáticas en la Facultad de Ciencias de la Universidad Nacional Autónoma de México. En el año 2007 ingresé al Programa de Doctorado en Ingeniería con especialidad en Ingeniería de Sistemas en la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León, trabajando bajo la dirección de la Dra. Ada Margarita Álvarez Socarrás y el Dr. Joaquín Pacheco Bonrostro en la tesis titulada "Un enfoque metaheurístico para un problema de ruteo con flexibilidad en las fechas de entrega".