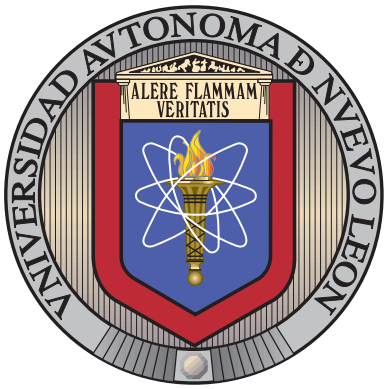
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado



Un problema bi-objetivo de ruteo de vehículos con ventanas de tiempo

por

Yadira Alondra De Santiago Badillo

en opción al grado de

Maestría en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León junio de 2011

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Un problema bi-objetivo de ruteo de vehículos con ventanas de tiempo

por

Yadira Alondra De Santiago Badillo

en opción al grado de

Maestría en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León junio de 2011

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 problema bi-objetivo de ruteo de vehículos con ventanas de tiempo », realizada por la alumna Yadira Alondra De Santiago Badillo, con número de matrícula 1508351, sea aceptada para su defensa como opción al grado de Maestría en Ciencias en Ingeniería de Sistemas.

El Comité de Tesis

Dra. Ada Margarita Álvarez Socarrás
Asesor

Dr. Francisco R. Angel Bello Acosta
Revisor

Dra. María Belén Melián Batista
Revisor

Vo. Bo.

Dr. Moisés Hinojosa Rivera
División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, junio de 2011

A mi familia, que siempre ha estado apoyandome en los buenos y malos momentos de mi vida, los quiero mucho.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Dios te agradezgo mi existir.

Agradezco a todos los profesores del Posgrado de Ingeniería de Sistemas, por los conocimientos que me han compartido. En especial a mi asesora de tesis, Dra. Ada Álvarez por el enorme apoyo brindado.

Agradezco a la Dra. Belén Melián, por el apoyo brindado durante mi estancia en Tenerife, España.

Gracias a mi revisor de tesis, el Dr. Francisco Angel Bello por su apoyo y conocimientos compartidos.

Agradezco a CONACYT por la beca de tiempo completo otorgada, por la beca mixta y por la beca del proyecto 61903.

Resumen

Yadira Alondra De Santiago Badillo.

Candidato para el grado de Maestro en Ciencias
en Ingeniería de Sistemas.

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

Título del estudio:

Un problema bi-objetivo de ruteo de vehículos con ventanas de tiempo

Número de páginas: 89.

Objetivos y método de estudio: En este trabajo se propone un modelo matemático a un problema de ruteo de vehículos con ventanas de tiempo. Dada una flotilla de vehículos, se desean encontrar las rutas diarias que visiten a un grupo de clientes dispersos geográficamente, los cuales establecen horarios específicos de atención.

Formulamos este problema mediante un modelo de ruteo con múltiples objetivos que surgen a partir de diferentes puntos de vista. Desde el punto de vista de costo (objetivo económico), se desea minimizar la distancia total recorrida; desde el punto de vista de equidad (objetivo social), se desea balancear las rutas en cuanto

al tiempo de recorrido. Este es un problema real que se presenta en una empresa en Gran Canaria, España.

Se presentan dos metodologías de solución a nuestro problema, la primera basada en una Búsqueda Dispersa cuya construcción inicial de las soluciones se realiza de forma secuencial y minimizando el objetivo de distancia, y la búsqueda local que se presenta en el proceso de mejora, tiene como objetivo minimizar el balance de tiempo, y el vecindario se define con el movimiento de reubicación entre rutas, es decir, un cliente es removido de su posición actual de una ruta e insertado en otra ruta siempre que sea factible respecto a las ventanas de tiempo.

La segunda metodología toma algunas ideas del MOAMP, la construcción de las soluciones se realiza de forma paralela y se implementan varias búsquedas locales, una de ellas siguiendo como función objetivo la suma ponderada de ambos objetivos.

Contribuciones y conlusiones: Este trabajo contribuye a dar solución a una problemática real, además que los objetivos a optimizar aún no han sido tratados en la literatura científica hasta ahora publicada. Por lo tanto, el modelo que se presenta y los algoritmos son también una contribución de este trabajo.

Firma del asesor: _____

Dra. Ada Margarita Álvarez Socarrás

Capítulo 1

Introducción

1.1 Descripción del problema

El presente trabajo fué motivado por una problemática que presenta una empresa en Las Palmas, Gran Canaria, España. La empresa cuenta con una flotilla de vehículos para transportar sus productos desde un depósito central hacia un conjunto de clientes distribuidos geográficamente. La demanda de cada cliente es conocida con anticipación, así como el intervalo de tiempo en que deben ser servidos. Se deben diseñar diariamente las rutas para la entrega de productos de manera que, por una parte se minimice la distancia total recorrida y por otra, se logre un equilibrio o balance en la carga de trabajo de los choferes. Este balance se ha definido en relación a la duración de las rutas, por lo que se desea entonces que la diferencia entre la ruta más larga y la ruta más corta sea lo menor posible. Cabe mencionar que las ventanas de tiempo para el servicio se consideran como restricciones duras, esto es, un vehículo no puede llegar a un cliente después del límite superior establecido por ese cliente. Sin embargo, si el vehículo llega antes del inicio del límite inferior, éste debe esperar.

Además de las restricciones relacionadas con las ventanas de tiempo, también están presentes restricciones referentes a la capacidad de los vehículos.

El problema que se aborda es entonces un problema bi-objetivo de ruteo de vehículos con ventanas de tiempo. Dado que el problema mono-objetivo de ruteo de

vehículos está clasificado como un problema combinatorio NP-difícil, el trabajo se dedica al desarrollo de algoritmos metaheurísticos que puedan ofrecer soluciones de calidad en tiempos aceptables para la empresa.

1.2 Hipótesis

Dada la naturaleza bi-objetivo del problema bajo estudio, aunado a la complejidad computacional del problema de ruteo de vehículos con ventanas de tiempo, se vuelve difícil obtener de forma exacta el conjunto de soluciones no dominadas. Por ello este trabajo se centra en proponer algoritmos que obtengan soluciones significativamente mejores que las que actualmente implementa la empresa, tanto en calidad como en el tiempo requerido para obtenerlas. La hipótesis principal es que los algoritmos que se proponen tienen un buen desempeño en comparación con las técnicas exactas.

1.3 Justificación

La relevancia del trabajo está dada principalmente por lo siguiente: por una parte el trabajo desarrollado contribuye a dar solución a una problemática real. Por otra parte, el problema de minimizar la distancia total recorrida conjuntamente con el balance de la carga de trabajo no ha sido tratado en la literatura científica hasta ahora publicada, por ello el estudio con respecto al tiempo desarrollado, el modelo propuesto y los algoritmos diseñados pueden ser aplicados y extendidos eficientemente en situaciones similares.

1.4 Objetivos

Los objetivos de este trabajo son los siguientes:

- Revisión de la literatura donde aborden problemas reales similares al nuestro.

- Modelación matemática de este problema.
- Estudio de las metodologías aplicadas a problemas multi-objetivo de ruteo de vehículos.
- Diseño de algoritmos de solución para el problema bi-objetivo basados en estrategias metaheurísticas.
- Implementación computacional de los algoritmos y realización de experimentos computacionales para medir el desempeño de los algoritmos propuestos.

1.5 Estructura de la tesis

La estructura de este trabajo es la siguiete: en el segundo capítulo se describen las características del problema de ruteo de vehículos capacitado y algunas de sus variantes más importantes; se comentan algunos métodos de solución tratados en la literatura para los problemas mono-objetivo, así como la descripción de los problemas de ruteo multi-objetivo que se han reportado y sus técnicas de solución.

En el capítulo tres se presenta el planteamiento del problema desde el enfoque bi-objetivo y se presenta el modelo matemático lineal entero mixto.

En el capítulo cuatro se describen dos metodologías que proponemos para dar solución al problema. La primera de ellas se basa en una búsqueda dispersa, adaptando algunos procedimientos a nuestro caso de dos objetivos. La segunda metodología retoma algunas ideas del MOAMP, una metodología reportada recientemente en la literatura para abordar los problemas multi-objetivo.

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

Finalmente, en el capítulo seis se presentan las conclusiones.

Capítulo 2

Antecedentes

El problema de ruteo de vehículos(VRP, por sus siglas en inglés) es uno de los principales problemas de la investigación de operaciones y también de optimización combinatoria, el cual tiene múltiples aplicaciones tanto en telecomunicaciones, como transporte, planeación de la producción, entre otros. El primer problema que se presentó fue en 1959 donde Dantzing y Ramser[12] describen una aplicación de la entrega de gasolina a las estaciones de servicio, proponiendo la primera formulación de programación matemática y un algoritmo de aproximación. Posterior a este trabajo se han presentado en la literatura muchas aplicaciones que involucran la entrega o recolección de algún producto o mercancía, cuyos objetivos suelen ser optimizar algún costo (distancia, tiempo, etc).

En la primera sección de este capítulo se inicia con la descripción del problema capacitado de ruteo de vehículos (CVRP), el cual es uno de los problemas más estudiados en la literatura así como el problema de ruteo de vehículos con ventanas de tiempo (VRPTW) que es una de sus principales variantes. En la sección 2.2 se describirán algunos de los métodos de solución para el problema de ruteo que se han abordado en la literatura del VRP mono-objetivo. En la sección 2.3 se hablará sobre los problemas multi-objetivo para el problema de ruteo de vehículos, describiendo más a detalle el problema multi-objetivo del ruteo de vehículos con ventanas de tiempo, pues es el tema a desarrollar en la tesis. Finalmente, en la sección 2.4 se describirán algunas de las metodologías de solución a los problemas multi-objetivo, abordando en esta sección las metodologías que se han propuesto en los últimos años

con enfoque multi-objetivo.

2.1 El problema básico de ruteo de vehículos y algunas variantes

2.1.1 El problema de ruteo de vehículos capacitado

El Problema de Ruteo de Vehículos Capacitado (CVRP)[40] es uno de los problemas básicos de ruteo de vehículos, el cual consiste en que a cada cliente se le debe enviar cierta cantidad de mercancía, ya conocida de inicio, desde un único depósito. Para repartirla se dispone de un determinado número de vehículos con cierta capacidad de producto a transportar, los cuales parten y llegan a un único depósito. Cada vehículo debe visitar exactamente a un único cliente, también cada cliente debe ser visitado por un solo vehículo y la suma de las demandas de los clientes visitados por un vehículo no deben exceder la capacidad del mismo. El objetivo de este tipo de problemas suele ser minimizar el costo total, donde dicho costo se puede ver como una función del tiempo o la distancia total recorrida, o bien minimizar el número de vehículos a utilizar.

Una recopilación de algoritmos de solución al problema de ruteo de vehículos capacitado se encuentra en los primeros capítulos del libro de Toth y Vigo[40].

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

2.1.2 Algunas variantes importantes del CVRP

Una variante interesante del CVRP es el CVRP con restricción de distancia (DCVRP), donde cada ruta está restringida a una distancia (o tiempo) máxima, además de la capacidad de los vehículos.

Otra variante es cuando se considera simultáneamente entrega y recolección de producto (VRP with Pickup and Delivery -VRPPD) [29], es decir, no sólo se requiere entregar mercancía de un depósito a los clientes, sino que además se debe recoger mercancía de algunos clientes hacia el depósito.

Una extensión importante del CVRP es cuando existe un tiempo de atención a cada cliente, esto es, un tiempo en que el cliente puede dar o recibir la mercancía o producto. Este problema se conoce en la literatura como problema de ruteo de vehículos con ventanas de tiempo (VRPTW), el cual se describe en la siguiente subsección por estar estrechamente relacionado al trabajo de esta tesis.

En [26] podemos encontrar los modelos y algoritmos de solución de los problemas anteriormente descritos.

2.1.3 Problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW)

Como ya se mencionó en la subsección anterior, existen muchas variantes del CVRP, sin embargo una extensión importante de dicho problema es el VRPTW en donde cada cliente i es servido por un vehículo en un intervalo de tiempo definido o ventana de tiempo, denotada como $[a_i, b_i]$. Para que un vehículo pueda dar sevicio a un cliente, éste debe llegar antes del inicio de la ventana de tiempo o dentro de la misma, pero si llegara después del fin de la ventana de tiempo, entonces ya no es posible brindar el servicio.

Cuando la restricción de la ventana de tiempo se debe cumplir, entonces se conoce como el caso duro o "hard", pues no se permite que el vehículo dé servicio al cliente después del limite superior de la ventana de tiempo. Mientras que cuando esta restricción puede no cumplirse, entonces estamos en el caso ligero o "soft". En los últimos años, se ha estudiado más el caso de las ventanas de tiempo soft por algunos autores como Taillard et al.[39].

El VRPTW es considerado un problema NP-difícil en sentido fuerte, pues contiene como un caso particular al CVRP, tomando $a_i = 0$ y $b_i = \infty$.

Algunos de los objetivos que se manejan en la literatura son el minimizar el número de vehículos utilizados para la repartición o recolección de mercancía, minimizar la distancia total recorrida, minimizar el tiempo total recorrido, entre otros.

Un resumen de algoritmos de solución al VRPTW lo encontramos en Braysy y Gendreau[6], o también podemos consultar el capítulo 7 de Toth y Vigo[40] donde se muestran modelos y algoritmos de solución.

Cabe mencionar que los problemas descritos anteriormente solo consideran un solo objetivo, sin embargo en la práctica muchas veces no sólo se desea optimizar el costo (distancia, tiempo o dinero), sino que se tienen varios objetivos simultáneamente. Por ejemplo, es posible considerar objetivos que a su vez optimizen un costo monetario y un costo biológico, si hablamos de alguna empresa que fabrica productos que contaminan en diferentes niveles al medio ambiente y desea fabricar el mayor número de productos de manera que se contamine lo menor posible al ambiente, pero que a su vez la empresa incremente lo más que se pueda sus ganancias con la elaboración de los productos [36]. Este tipo de problemas con varios objetivos será descrito más adelante en las secciones 2.3 y 2.4

2.2 Metodologías de solución al VRP

mono-objetivo

Primeramente mencionaremos algunos de los métodos de solución al Problema de Ruteo de Vehículos Capacitado, tanto métodos exactos como heurísticos y posteriormente se mencionarán algunas metaheurísticas.

2.2.1 Algoritmos exactos de solución

Algoritmos exactos

Dentro de los algoritmos exactos de solución al VRP tenemos los siguientes:

Ramificación y acotamiento (Branch and Bound): Hasta finales de 1980, los algoritmos más efectivos para el VRP eran los de ramificación y acotamiento. Recientemente, se han propuesto relajaciones más sofisticadas como las relajaciones Lagrangianas o el procedimiento *additive bounding*, con el cual se ha incrementado el tamaño de los problemas a resolverse de forma óptima, para más detalles ver Toth y Vigo [40].

Ramificación y corte (Branch and Cut): Es considerado, hasta la fecha, el mejor algoritmo exacto para el VRP. Este algoritmo utiliza la relajación lineal del modelo de Programación Lineal Entera (IP), es decir, al problema de programación lineal entera relajan la condición de que todas las variables sean enteras. En el capítulo 3 de Toth y Vigo (2002) se encuentra explicado a detalle.

2.2.2 Algoritmos heurísticos de solución al VRP

Distintas familias de heurísticas se han propuesto para el Problema de Ruteo de Vehículos. A grandes rasgos estas se pueden clasificar en dos clases: *heurísticas clásicas*, desarrolladas entre 1960 y 1990, y *Metaheurísticas*, cuyo crecimiento ha sido en la última década.

Heurísticas clásicas

Las heurísticas clásicas realizan una exploración limitada del espacio de búsqueda y típicamente producen soluciones buenas en poco tiempo. Su implementación es relativamente simple y pueden ser fácilmente adaptadas para incluir las restricciones encontradas en aplicaciones reales.

Estas heurísticas pueden clasificarse como se menciona en Zeng [41] en las siguientes tres categorías:

1. Heurísticas Constructivas.
2. Heurísticas de dos fases.
3. Algoritmos de mejora o búsqueda local.

A continuación se explica cada una de ellas.

1. **Heurísticas Constructivas**: La solución se va creando al agregar componentes repetidamente a una solución vacía hasta que esté completa. Típicamente se pueden clasificar en:

   a) De insercción: parten de una solución con rutas vacías y en cada paso insertan un cliente nuevo a las rutas existentes.

   b) De combinación: se crea una solución que tenga una ruta para cada cliente y luego se van uniendo repetidamente las rutas.

   Generalmente la elección de qué componente se modifica en cada caso responde a una estrategia golosa, es decir, se escoge el mejor componente disponible en el momento de la decisión. Debido a la compleja naturaleza del problema puede resultar posteriormente que esa decisión no fue la mejor.

   En la mayoría de los casos las heurísticas constructivas pueden implementarse de dos maneras:

   ♢ Secuenciales: trabajan sobre una ruta a la vez, cuando ya no es posible realizar ningún otro cambio se continúa con la siguiente.

   ♢ Paralelas: actúan sobre todas las rutas simultáneamente y en cada iteración se modifica la ruta más conveniente. Generalmente requieren más cálculo pero suelen producir mejores resultados.

Las heurísticas constructivas más conocidas para el VRP son la de Clarke-Wright [25] y Mole y Jameson [27].

En el algoritmo de Clarke-Wright, también conocido como el algoritmo de los ahorros, el número de vehículos no está determinado, y el método inicia con rutas que contienen el depósito y otro vértice, en cada paso se van combinando rutas si es que se genera el mayor ahorrro posible.

Supongamos que se tienen las rutas $(0, \ldots, i, 0)$ y $(0, j, \ldots, 0)$ y que queremos unirlas en una sola ruta $(0, \ldots, i, j, \ldots, 0)$, es decir, queremos formar la ruta que parte del depósito, llega al nodo i y en lugar de regresar al depósito ahora visita al nodo j y sigue su recorrido hasta llegar nuevamente al depósito. Es conveniente fusionar estas rutas en una sola ruta factible si al calcular la distancia de ahorro $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, ésta resulta en un ahorro positivo. En la Figura 2.1 podemos observar en el lado izquierdo las dos rutas iniciales, donde el cuadrado representa al depósito, y las dos rutas que consideramos inicialmente considerando al nodo i y j, y en la figura del lado derecho observamos una sola ruta después de unir las dos rutas dadas. Este algoritmo puede ser aplicado para la construcción secuencial y en paralelo

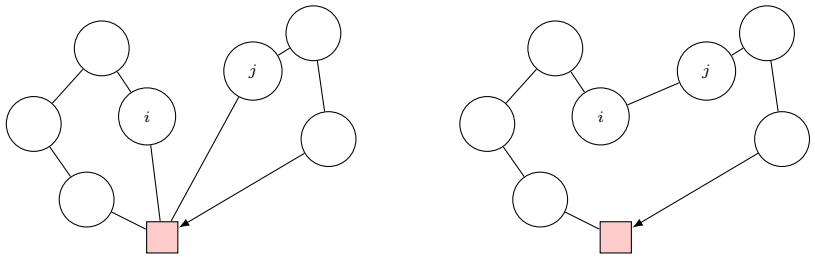


Figura 2.1: Algoritmo de Clarke-Wright

La heurística de construcción de Mole y Jameson, descrita en [40], utiliza como criterio de selección e inserción la evaluación de la distancia que resulta al insertar en una ruta a un cliente k no ruteado entre dos clientes consecutivos i y j, es decir, $\alpha(i, k, j) = c_{ik} + c_{kj} - \lambda c_{ij}$, donde λ es un parámetro controlado

por el usuario.

Una heurística más general es la que se menciona en Laporte [24], propuesta por Christofides et al. En el primer paso, se utiliza el algoritmo de inserción secuencial para determinar un conjunto de rutas factibles. Y el segundo paso hace uso del algoritmo de inserción en paralelo. Para cada ruta determinada en el primer paso, se selecciona un cliente e inicializamos para cada uno de estos una ruta. Los clientes no ruteados se insertan utilizando el criterio de peso, es decir, la diferencia entre la mejor y la segunda mejor inserción, y cada ruta parcial que se va construyendo se mejora con un procedimiento de búsqueda local, 3-opt explicado posteriormente. El algoritmo descrito representa un mejor compromiso entre eficacia y eficiencia.

2. **Heurísticas de dos fases**: el problema se descompone en sus dos componentes naturales, agrupar los clientes en rutas factibles y construir la ruta. Las heurísticas de dos fases se dividen en dos clases:

   a) Cluster-first, route-second: los clientes se organizan en grupos, se les asigna un vehículo y se construye una ruta para cada grupo.

   b) Route-first, cluster-second: se construye una solución que visite a todos los clientes la cual se fracciona luego en distintas rutas factibles. Debido a las ventanas de tiempo es de suma importancia para nuestro problema.

   El algoritmo de Fisher y Jaikumar (FJ) [15], es uno de los más conocidos en la clase de cluster-first route-second. En la primer fase, los clientes se reparten en m grupos resolviendo un problema adecuado de asignación generalizado (GAP por sus siglas en inglés), donde un cluster denota un conjunto de clientes visitado por un vehículo. Se selecciona un cliente semilla r_k para cada cluster o ruta k, y se calcula el costo de asignar el cliente j al cluster k como $d_{jk} = min\{c_{0,j}+c_{j,r_k}+c_{r_k,0}, c_{0,r_k}+c_{r_k,j}+c_{j,0}\}-(c_{0,r_k}+c_{r_k,0})$. En la segunda fase, se determina la ruta optima para cada cluster, resolviendo el correspondiente TSP.

3. **Algoritmos de mejora o búsqueda local**: parten de una solución inicial e intentan mejorarla realizando pequeños cambios como reubicar o intercambiar clientes, invertir secuencias de clientes, entre otros. Cada método define un vecindario que contiene a aquellas soluciones que se pueden obtener a partir de la actual. En ese vecindario se busca una mejor solución y al encontrarla, ésta remplaza a la actual. La búsqueda se repite hasta que se llega al punto en que ya no se puede mejorar la solución actual. Se dice entonces que se ha llegado a un óptimo local.

   Los movimientos para mejorar una ruta factible más utilizados en la literatura son movimientos entre-rutas y movimientos dentro-rutas, o como nos referiremos de aquí en adelante inter-ruta [39] e intra-rutas.

   Los movimientos inter-rutas consisten en mover vértices o cadenas de vértices de una ruta a otra. En la Figura 2.2 podemos observar el movimiento de recolocación de una cadena simple (un nodo), dadas las rutas $(0, \ldots, i-1, i, i+1, \ldots, 0)$ y $(0, \ldots, j-1, j, \ldots, 0)$. En el lado izquierdo de la figura se observan las rutas iniciales y del lado derecho tenemos que al recolocar al nodo i después del nodo $j-1$ obtenemos las nuevas rutas $(0, \ldots, i-1, i+1, \ldots, 0)$ y $(0, \ldots, j-1, i, j, \ldots, 0)$.

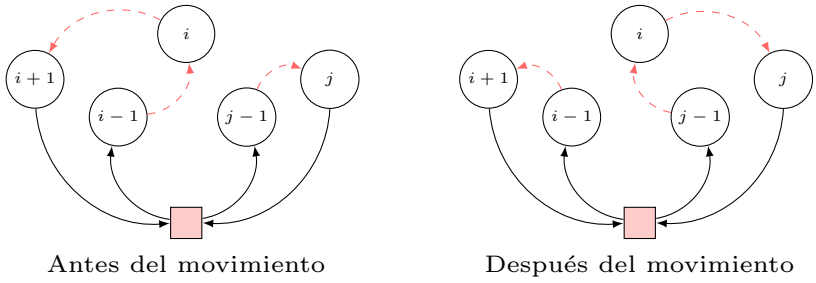


Figura 2.2: Movimiento inter-ruta: Relocolación simple

Otro movimiento muy utilizado entre rutas es CrossOver. Consiste en dividir cada una de las rutas seleccionadas en dos secciones, (lo cual se logra eliminando un arco de cada ruta) e insertando dos nuevos arcos que conecten,

respectivamente, la sección inicial de la primer ruta con la sección final de la segunda y viceversa. Para ejemplificar el movimiento CrossOver, supongamos que tenemos las rutas $(0, \ldots, i-1, i, \ldots, 0)$ y $(0, \ldots, j-1, j, \ldots, 0)$ graficadas en la Figura 2.3 y supongamos que los arcos a eliminar son $(i-1, i)$ y $(j-1, j)$, del lado derecho de la figura se observan las rutas que se obtienen al unir las rutas agregando los arcos $(j-1, i)$ y $(i-1, j)$ de las rutas iniciales, es decir, las nuevas rutas son $(0, \ldots, j-1, i, \ldots, 0)$ y $(0, \ldots, i-1, j, \ldots, 0)$.

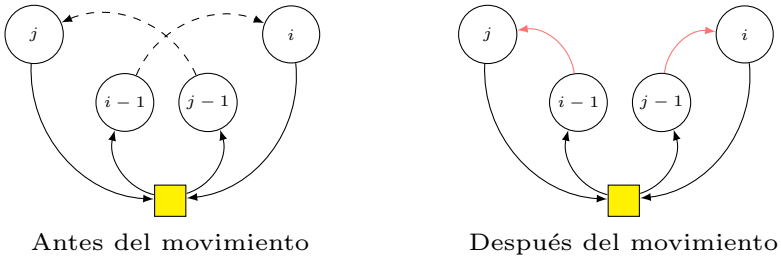


Figura 2.3: Movimiento inter-ruta: CrossOver

En el caso de movimientos intra-rutas los más utilizados en la literatura son el procedimiento k-opt y Or-opt. En especial, cuando $k = 2$, tenemos el 2-opt, que consiste en quitar dos arcos no adyacentes de una ruta y agregar dos nuevos arcos que mejoren la solución y reconecten la ruta. En la Figura 2.4 se ejemplifica el movimiento. Del lado izquierdo tenemos la ruta $(0, \ldots, i-1, i, j-1, j, \ldots, 0)$, y supongamos que los arcos a remover son $(i-1, i)$ y $(j-1, j)$, entonces la nueva ruta se obtiene al agregar los arcos $(i-1, j)$ e (i, j), quedando la ruta $(0, \ldots, i-1, j-1, i, j, \ldots, 0)$, la que se muestra del lado derecho de la figura.

El movimiento Or-opt, propuesto por Or en 1976, considera la cadena de uno, dos o tres clientes consecutivos de una solución y la mueve a otro lugar dentro de la misma ruta (aunque también se puede aplicar a movimientos inter-rutas).

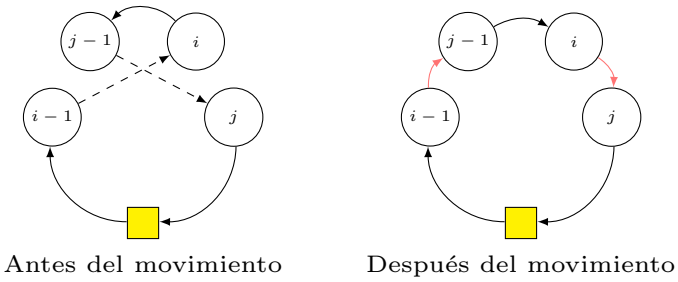


Figura 2.4: Movimiento intra-ruta: 2-opt

2.2.3 Metaheurísticas de solución al VRP

Las metaheurísticas son un conjunto de conceptos algorítmicos que pueden ser usados para definir métodos aplicables a un gran número de problemas distintos. No son exclusivas del problema de ruteo de vehículos, los mismos conceptos pueden aplicarse a muchos otros problemas combinatorios.

Una metaheurística puede entenderse como un procedimiento diseñado para guiar a otras heurísticas subyacentes, específicas para el problema de construcción y búsqueda local, hacia regiones prometedoras del espacio de búsqueda.

Así las metaheurísticas realizan una exploración más profunda en el espacio de búsqueda con el fin de encontrar soluciones de mayor calidad pero a expensas de un mayor tiempo de procesamiento. A pesar de ser marcos algorítmicos que pueden ser aplicados a diferentes problemas de optimización requieren generalmente el ajuste de muchos parámetros y decisiones en su diseño.

La gran diferencia respecto de las heurísticas clásicas es que muchas permiten el empeoramiento de la solución o incluso soluciones intermedias no factibles en el proceso de búsqueda. Todo con el objetivo de escapar de un mínimo local.

La búsqueda local es una de las componentes importantes de muchas metaheurísticas, por lo que primero definiremos algunos conceptos genéricos:

Definición 2.1 *Sea X el conjunto de soluciones del problema combinatorio. Cada solución x tiene un conjunto de soluciones asociadas $N(x) \subseteq X$, que denominaremos entorno o vecindad de x.*

Definición 2.2 *Dada una solución x, cada solución de su vecindad, $x' \in N(x)$, puede obtenerse directamente a partir de x mediante una operación llamada movimiento.*

Entonces, una búsqueda local se basa en explorar la vecindad de una solución y seleccionar una nueva solución en ella, es decir, realizar el movimiento asociado. Desde la nueva solución se explora su vecindad y se repite el proceso.

Dependiendo de la estructura del problema a resolver, se define la vecindad y movimiento que se debe realizar para aplicar la búsqueda local.

Algunas de las metaheurísticas utilizadas para el VRP son:

1. *Búsqueda tabú (Tabu Search-TS-)*
2. *Algoritmos Genéticos (Genetic Algorithms-GA-)*
3. *Búsqueda Dispersa (Scatter Search)*
4. *Colonia de Hormigas (Ant Colony)*

A continuación se da una breve descripción de las metaheurísticas anteriores.

Búsqueda Tabú

La Búsqueda Tabú es la metaheurística más utilizada, cuya principal característica es el uso de memoria adaptativa, en Glover y Laguna [18] podemos encontrar numerosas aplicaciones. TS se mueve iterativamente de una solución x_1 a otra x_2 en la vecindad de x_1, es decir, en $N(x_1)$, sin embargo en lugar de considerar toda la vecindad o entorno de soluciones, la búsqueda se define en un entorno

reducido. Una forma de definir este entorno reducido consiste en etiquetar a las soluciones ya visitadas como tabú, las cuales son almacenadas en una lista tabú por un determinado número de iteraciones (parámetro definido por el usuario) con el fin de evitar que se vuelvan a visitar estas soluciones o bien evitar ciclos.

En [11] podemos encontrar una recopilación de la aplicación de la búsqueda tabú para el problema de ruteo de vehículos.

Algoritmos Genéticos

Este algoritmo trata de una técnica inspirada en la evolución biológica. En cada iteración se analiza una población de soluciones. Los individuos más aptos, aquellos con mejor valor en la función objetivo, tendrán mayor posibilidad que sus descendientes formen parte de la nueva población mientras que los menos aptos son descartados. Las nuevas generaciones se originan del proceso de mutación y recombinación genética de la población antigua.

En Braysy et al.[6] se reporta una aplicación del algoritmo al problema de ruteo de vehículos con ventanas de tiempo, y en [7] tenemos una recopilación de la aplicación de este algoritmo.

Búsqueda Dispersa

Esta metaheurística fue desarrollada por Glover[18], opera sobre un conjunto de soluciones, llamado *conjunto de referencia (RefSet)*. Es una metaheurística evolutiva, lo cual se logra a través de la combinación de las soluciones del conjunto $RefSet$, este conjunto está formado de "buenas soluciones", donde no solo se limita a tener soluciones de calidad respecto a la función objetivo sino que también considera soluciones diversas con respecto a las otras soluciones en el conjunto de referencia.

En términos generales la implementación de una búsqueda dispersa se compone de los siguientes métodos o procedimientos:

1. Un generador de soluciones diversas.
2. Un método de mejora.
3. Un método para crear y actualizar el conjunto de referencia $RefSet$.
4. Un método para generar subconjuntos de $RefSet$ a los que se aplicará el método de combinación.
5. Un método de combinación.

En [37] encontramos una aplicación de Búsqueda Dispersa al problema de ruteo de vehículos con ventanas de tiempo.

Colonia de Hormigas (Ant Colony)

Los Sistemas de Hormigas o Ant Systems se inspiran en la estrategia utilizada por las colonias de hormigas para buscar alimentos. Cuando una hormiga encuentra un camino hacia una fuente de alimento, deposita en el trayecto una sustancia llamada $feromona$. La cantidad de feromona depositada depende de la longitud del camino y de la calidad del alimento encontrado. Si una hormiga no detecta la presencia de feromona se mueve aleatoriamente; pero si percibe la sustancia, decidirá con alta probabilidad moverse por los trayectos que contengan la mayor cantidad de sustancia, lo que a su vez provocará un incremento en la cantidad de feromona en ese trayecto.

Para ver una aplicación de esta metaheurística se puede consultar [8].

2.3 Problemas multi-objetivo de ruteo de vehículos

Como se ha comentado, los problemas de ruteo de vehículos y sus variantes han sido ampliamente abordados en la literatura, pues estos han sevido para modelar

varios casos de la vida real. Generalmente los problemas que se han modelado han sido con el fin de optimizar un solo objetivo que usualmente es minimizar algún costo. Sin embargo, en muchos casos no solo es importante optimizar un costo, sino que hay otros objetivos a considerar, como puede ser minimizar el número de vehículos a utilizar o minimizar el tiempo de espera de los clientes, etc. Entonces, al considerar más de un objetivo tendremos un problema multi-objetivo.

Algunos de los problemas de ruteo multi-objetivo no comparten los mismos objetivos, por ejemplo Borgulya [5] considera minimizar: la distancia recorrida por los vehículos y la diferencia entre la ruta más larga y la ruta más corta, mientras que los objetivos de Murata e Itai [30] son minimizar tanto el número de vehículos como el tiempo máximo recorrido por los vehículos.

2.3.1 Problema Bi-objetivo de Ruteo de Vehículos con Ventanas de Tiempo

En especial, cuando hablamos de problemas de ruteo de vehículos con ventanas de tiempo, podemos considerar por ejemplo un problema cuyos objetivos sean minimizar el tiempo de espera de los clientes para ser servidos dentro de un intevalo o ventana de tiempo y minimizar la distancia total recorrida por los vehículos, es decir, estaríamos hablando de un problema bi-objetivo con ventanas de tiempo. Este tipo de problemas han sido menos abordados en la literatura, podemos citar por ejemplo a Hong y Park[19], quienes consideran minimizar el tiempo total del recorrido de los vehículos y minimizar el tiempo total de espera de los clientes, o bien a Najera y Bullinaria [16] cuyos objetivos son minimizar tanto el número de rutas como el costo total.

Sin embargo, a diferencia de los trabajos mencionados anteriormente, los objetivos de este trabajo de tesis son minimizar la distancia total recorrida por los vehículos y minimizar la diferencia entre la ruta más larga y la ruta más corta en tiempo, objetivos que actualmente no se han abordado en ningún artículo publicado

a la fecha.

2.4 Metodologías de solución para problemas multi-objetivo

Los problemas en el mundo real, suelen por naturaleza tener más de un objetivo (posiblemente en conflicto entre sí) los cuales se desean optimizar de manera simultánea. Sin embargo estos problemas se suelen transformar en un problema mono-objetivo, donde todos los objetivos, excepto uno, se suelen agregar como restricciones adicionales.

Ahora bien, para dar solución a estos problemas multi-objetivo una técnica muy utilizada en la literatura es transformar el problema original, ponderando las funciones objetivo y sumándolas para dar lugar a un problema de optimización con un solo objetivo. Dentro de las desventajas de esta técnica es que para dar solución al problema de forma exacta el problema mono-objetivo resultante probablemente sigue siendo NP-Completo y además se requiere resolver múltiples problemas mono-objetivo, cada uno de ellos con ponderaciones diferentes. Otra desventaja de esta técnica es la dificultad de definir un conjunto de pesos que permita generar una porción importante de soluciones (Frente de Pareto, que se define más adelante).

Una clasificación para resolver los problemas multi-objetivo de ruteo de vehículos, de acuerdo a [21] es la siguiente:

a) Métodos escalares.

b) Métodos de Pareto.

c) Métodos que no se clasifican en ninguno de los anteriores.

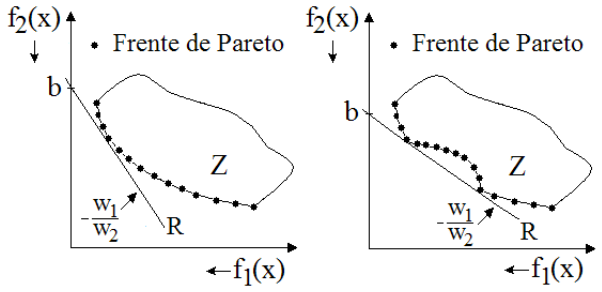


Figura 2.5: Representación gráfica del método de sumas ponderadas para una región convexa y del lado derecho una región no-convexa

2.4.1 Métodos escalares

En esta categoría se agrupan los métodos que usan alguna transformación matemática para trasformar el problema a uno con un solo objetivo.

El método más popular es el que ya hemos mencionado anteriormente, el **método de agregación o sumas ponderadas**. Sin embargo, a pesar de ser una técnica muy sencilla de implementar, otra desventaja que presenta es que se requiere convexidad en la región a trabajar para garantizar encontrar todos los puntos no dominados. En la Figura 2.5 del lado izquierdo se representa la recta R que corresponde a un vector de pesos (w_1, w_2) fijado. Z representa el conjunto de soluciones realizables en el espacio de dos objetivos $f_1(X)$ y $f_2(X)$. En el lado derecho de la figura se representa el caso cuando el frente de Pareto contiene regiones no convexas.

Otra técnica escalar utiliza los métodos de la **programación por metas**. Consiste en elegir una meta, es decir un punto en el espacio objetivo, y se dirige la búsqueda para minimizar la distancia entre el punto actual y la meta. Una dificultad de éste método es definir la meta. Este método ha sido utilizado por Hong y Park [19], quienes diseñan una heurística de dos fases. En la primera fase se crean conjuntos de clientes agrupados de acuerdo a las restricciones y a las preferencias que da el centro decisor. En la segunda fase se determina la ruta del vehículo para cada grupo.

Otro trabajo donde se utiliza esta técnica es [20]. Aquí los autores definen dinámicamente la meta y utilizan búsqueda tabú para llegar a ella. En este método se definen varias búsquedas tabú, cada una de ellas inicia desde la solución previamente encontrada. Para cada búsqueda se define una función objetivo, la cual debe tener encuenta la multiplicidad de las búsquedas locales, con el fin de evitar que dos búsquedas exploren la misma área del espacio objetivo. Con esto se intenta explorar todo el espacio objetivo y converger hacia el frente de Pareto.

También un problema multi-objetivo puede transformarse en uno mono-objetivo, utilizando el método de la **ϵ-restricción**. En esta técnica se optimiza un solo objetivo y los restantes se consideran como restricciones, las cuales se expresan como $f_i(x) \leq \epsilon_i$. Para encontrar el conjunto de soluciones de Pareto óptimas, se resuelve varias veces el problema multi-objetivo modificado, eligiendo diferentes valores ϵ_i. En la Figura 2.6 podemos observar cómo funciona este método de forma gráfica, suponiendo que el objetivo a optimizar es $f_1(x)$ y $f_2(x)$ es restringido por ϵ_2. Z representa el conjunto de soluciones realizables en el espacio de dos objetivos $f_1(X)$ y $f_2(X)$.

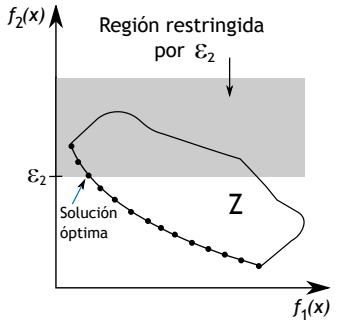


Figura 2.6: Representación gráfica del método de ϵ-restricciones

A pesar de ser una técnica que puede encontrar soluciones de Pareto que pertenecen a regiones no convexas, la carga computacional es considerable, además se pudiera elegir un valor ϵ para el cual no existe solución realizable, y también puede ocurrir que al elegir valores superiores a un cierto ϵ se encuentre siempre la misma solución realizable. En la Figura 2.7 se muestran ambos casos, por un lado para

valores $\epsilon_2 < \epsilon_2^t$ no contiene soluciones factibles y por otro cuando elegimos $\epsilon_2 > \epsilon_2^u$ siempre encontramos la solución u.

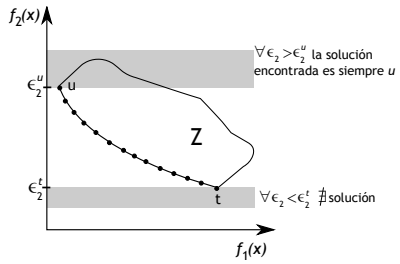


Figura 2.7: Representación gráfica de algunos inconvenientes del método de las ϵ-restricciones

Aplicaciones de esta técnica podemos encontrar en Corberán et al [10], los cuales utilizan la búsqueda dispersa (SS) para resolver cada problema. La misma estrategia ha sido utilizada por Pacheco y Martí [33], pero en lugar de SS utilizan Búsqueda Tabú.

2.4.2 Métodos de Pareto

En esta categoría se agrupan los métodos que utilizan directamente la noción de dominancia de Pareto. Esta metodología fue introducida principalmente para los algoritmos genéticos, muchos autores han usado algoritmos evolutivos con métodos de Pareto para resolver problemas de ruteo multi-objetivo. Por ejemplo Ombuki et al. [32] aplica esta técnica usando algoritmos genéticos y cuyos objetivos son minimizar el número de venículos y el costo total al problema de ruteo de vehículos con ventanas de tiempo. Para más aplicaciones sobre este método consultar [21].

2.4.3 Métodos no escalares y no Pareto

Cuando no se utilizan técnicas escalares ni métodos de Pareto, otra técnica que se reporta en la literatura son las:

a) basadas en algoritmos genéticos, donde Vector Evaluated Genetic Algorithm (VEGA), fue el primer algoritmo genético utilizado para resolver problemas multi-objetivo;

b) estrategias lexicográficas, en donde a cada objetivo se le da un valor de prioridad y los problemas se resuelven en orden decreciente de dicha prioridad dada;

c) mecanismos de colonias de hormigas o

d) alguna otra heurística específica.

Algoritmos Genéticos

Como se mencionaba anteriormente VEGA es considerado como el primer algoritmo evolutivo diseñado para la resolución de problemas multi-objetivo [14]. La diferencia con un algoritmo genético simple es la manera de efectuar la selección. VEGA considera una población de N individuos y si se tienen k objetivos, una selección por torneo de $\frac{N}{k}$ individuos es realizada para cada función objetivo. De manera, que se crean k poblaciones, de forma que cada una de ellas contiene los $\frac{N}{k}$ mejores individuos para una función objetivo k. Posteriormente se mezclan las k subpoblaciones y se termina con la aplicación de los operadores genéticos de cruce y mutación. Doerner et al.[14] utilizan VEGA para generar conjuntos de nodos visitados.

Métodos lexicográficos

Este método es de interés cuando se tiene una clara jerarquía entre las diferentes funciones objetivo. Así cuando se optimiza para el primer objetivo, su valor no cambia y se agrega una nueva restricción al problema. Para ver aplicaciones del método a otros problemas puede consultar [21].

2.4.4 Metaheurísticas para problemas multi-objetivo

Para la resolución de problemas multi-objetivo una de las metaheurísticas más utilizadas son los algoritmos evolutivos, los cuales manipulan una población de soluciones, a diferencia de la mayoría de otros métodos que operan sobre una única solución. Esta diferencia proporciona a los algoritmos evolutivos multi-objetivo la habilidad de encontrar múltiples soluciones del frente de Pareto, ya sea para espacios continuos, no continuos, discretos, no convexos, etc.

Un algoritmo evolutivo para resolver problemas multi-objetivo es el algoritmo genético de ordenamiento no dominado **NSGA-II** (Nondominated Sorting Genetic Algorithm II)[13], es un algoritmo elitista, de manera que en cada nueva generación los mejores individuos encontrados son conservados. Además, el algoritmo integra un operador de selección basado en "*crowding distance*", la cual evalúa la densidad de soluciones paresentes alrededor de una solución i, por ejemplo para una solución i intermedia (en el frente de Pareto), esta distancia se calcula en función del perímetro del hibercubo de los vértices a las soluciones más próximas de i sobre cada objetivo. Y la asignación de este valor disminuirá las posibilidades de supervivencia de una solución i presente en una región donde se concentran las soluciones. Por ejemplo en la Figura 2.8 la solución i será eliminada dado que el perímetro de su cuboide es el más pequeño.

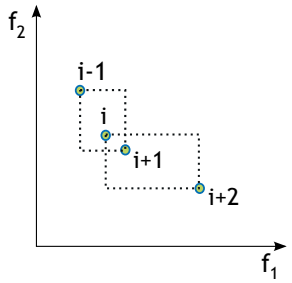


Figura 2.8: Distancia de agrupamiento (*crowding distance*) utilizada en NSGAII

Un esquema para el procedimiento de NSGAII se muestra en la Figura 2.9. Supongamos que en una generación t tenemos las poblaciones P^t y Q^t de tamaño

N, y sea P^t la población que contiene los mejores individuos encontrados hasta esta generación y Q^t formada por los individuos resultantes de las fases anteriores del algoritmo (selección, cruce y mutaciíon). Lo primero que se realiza es generar la población $R^t = P^t \cup Q^t$, la cual se clasifica por rangos de acuerdo a la no-dominancia, para así identificar los diferentes frentes de soluciones no dominadas, donde los mejores individuos se encuentran en los primeros frentes. Después se construye una nueva población P^{t+1}, la cual contiene elementos de frentes completos, por ejemplo añadir el frente F_1 de rango 1, el segundo frente F_2 de rango 2, etc. sin sobrepasar el tamaño N. Sin embargo, si el número de individuos presentes en P^{t+1} es menor que N, el operador *crowding distance* es aplicado sobre el siguiente primer frente F_i, que aún no se ha incluido en P^{t+1}. Este operador lo que hace es introducir los mejores $(N - |P^{t+1}|)$ individuos al conjunto P^{t+1}. Finalmente, los operadores de selección, cruce y mutación son aplicados sobre los individuos de P^{t+1}, a fin de generar la población de descendientes Q^{t+1}.

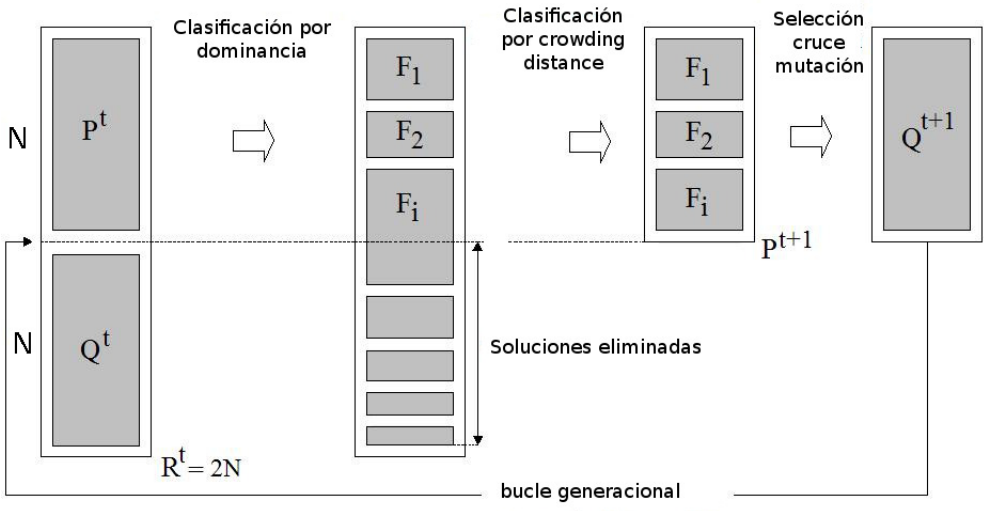


Figura 2.9: Esquema del procedimiento del NSGAII

Otros algoritmos evolutivos aplicados a los problemas multi-objetivo son SPEA2[42], o bien PAES[23], entre otros.

La búsqueda dispersa ($scattersearch - SS$) es una algoritmo metaheurístico

que también es considerado como un algoritmo evolutivo, pues en esencia incorpora el concepto de evolucionar una población. En los últimos anõs se ha tomado más en cuenta para extenderlo a los problemas multi-objetivo.

Por ejemplo Beausoleil [3] propone MOSS, que es un algoritmo que utiliza un método híbrido de búsquedas tabu/scatter-search para resolver problemas de optimización multi-objetivo no lineales. Hace uso de la búsqueda tabú en el método de generación de diversificación para generar soluciones diversas en la aproximación del conjunto de Pareto, además en la actualización del conjunto de referencia también se utiliza la búsqueda tabú.

El método desarrollado por Caballero et al. [9], es un método para optimización multi-objetivo basado en una serie de búsquedas tabú enlazadas, denominado MOAMP (MultiObjective Adaptive Memory Procedure), por sus siglas en inglés y como haremos referencia de aquí en adelante. Este método tratará de adaptar una búsqueda tabú a la estructura particular del conjunto eficiente en un problema multi-objetivo. Tiene la característica que la aproximación final del conjunto eficiente puede incluir cualquier solución visitada durante la búsqueda. 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, también es eficiente.

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

Otra metodología que se engloba en una búsqueda dispersa es la que proponen

Molina et al. [28] SSPMO, la cual se aplica a variables continuas consiste en una búsqueda híbrida scatter-search/tabu que incluye dos fases: la generación del conjunto inicial de puntos eficientes por medio de búsquedas tabú y la combinación de soluciones y actualización de la aproximación del frente de Pareto utilizando búsquedas dispersas. Para generar el conjunto de referencia SSPMO elige las mejores soluciones del conjunto inicial para cada uno de las funciones objetivo. El resto del conjunto de referencia se obtiene utilizando el criterio de max-min, que consiste en maximizar la mínima distancia de las soluciones que aún no se han agregado al conjunto de referencia y las que ya están.

Nebro et al. [31] propone un algoritmo híbrido denominado AbYSS (Archive-Based hYbrid Scatter Search), sigue una estructura de búsqueda dispersa, utiliza de los algoritmos evolutivos los operadores de mutación y cruce.

Capítulo 3

Descripción y modelación del problema

En este capítulo se describirá el problema de ruteo de vehículos con ventanas de tiempo que se abordará en este trabajo, así como un modelo matemático del mismo.

El problema de ruteo de vehículos con ventanas de tiempo, VRPTW por sus siglas en inglés y como nos referiremos de aquí en adelante, es un problema de diseñar rutas para una flota de vehículos idénticos al menor costo desde un depósito hacia un conjunto de puntos distribuidos geográficamente. Las rutas deben diseñarse de manera que cada punto sea visitado solamente por un vehículo dentro de un intervalo de tiempo dado para cada punto. Todas las rutas inician y terminan en el depósito, y la demanda total de todos los puntos de una ruta en particular no debe exceder la capacidad del vehículo.

Una descripción formal del VRPTW es:

Sea $G = (V, A)$ un grafo completo, donde $V = \{0, 1, \ldots, n\}$ es el conjunto de vértices y A es el conjunto de arcos. El subconjunto de vértices $I = \{1, \ldots, n\}$ corresponde a los clientes, mientras que el vértice 0 corresponde al depósito o almacén. Cada cliente i tiene asociado una demanda d_i, un tiempo de servicio s_i y una ventana de tiempo para iniciar el servicio $[a_i, b_i]$. Cada arco $(i, j)\ \epsilon\ A$ tiene asociado dos valores no negativos c_{ij} y t_{ij} que representan el costo y el tiempo de

transporte respectivamente de ir del vértice i al vértice j. El depósito también tiene asociada una ventana de tiempo $[a_0, b_0]$, donde a_0, representa la cota inferior para el tiempo de salida de los vehículos del depósito y b_0, la cota superior para el tiempo de llegada de los vehículos al depósito. Generalmente a_0 se toma igual a cero y b_0 igual al tiempo máximo (T) que pueden estar los vehículos fuera del depósito.

El problema consiste en determinar un conjunto de circuitos simples (rutas de vehículos) con el objetivo, generalmente, de minimizar la suma de los costos de todas las rutas y además, que se cumplan las siguientes condiciones:

a) Cada ruta debe contener el vértice 0.

b) Cada vértice $i \ \epsilon \ I$ debe ser visitado exactamente por una ruta.

c) La suma de las demandas de los vértices de una ruta no puede sobrepasar la capacidad del vehículo.

d) Para cada vértice $i \ \epsilon \ I$ su servicio debe comenzar dentro de su ventana de tiempo.

En este trabajo además de minimizar la distancia total, se considera un segundo objetivo relacionado con el balanceo de la duración en tiempo entre las diferentes rutas, es decir, minimizar la diferencia entre la duración de la ruta más larga y la ruta más corta.

El modelo bi-objetivo que se desarrollará está basado en un modelo de dos índices para las variables asociadas a los arcos y donde el número de rutas no está fijado con anticipación.

3.1 Parámetros

Los parámetros para el VRPTW son los siguientes:

n : Cantidadd de clientes.

$V = \{ 0, 1, \ldots, n \}$: Conjunto de índices para los nodos de la red, donde el índice 0 corresponde al depósito y el subconjunto $V \supset I = \{ 1, 2, \ldots, n \}$ es el conjunto de índices para los clientes.

d_i : Demanda del cliente i.

c_{ij} : Costo por utilizar el arco (i, j).

Q : Capacidad de los vehículos.

t_{ij} : Tiempo en recorrer el arco (i, j).

s_i : Tiempo de servicio del cliente i, donde para el depósito $s_0 = 0$.

a_i : Límite inferior de la ventana de tiempo para iniciar el servicio del cliente i.

b_i : Límite superior de la ventana de tiempo para iniciar el servicio del cliente i.

$a_0 = 0$: Cota inferior para el tiempo salida de los vehículo del depósito.

$b_0 = T$: Cota superior para el tiempo de llegada de los vehículos al depósito. Además representa la duración máxima que puede tener una ruta.

3.2 Variables del problema

Para modelar el problema de ruteo de vehículos con ventanas de tiempo necesitamos dos tipos de variables: de decisión y auxiliares, definidas en las siguientes subsecciones:

3.2.1 Variables de decisión

La variable de decisión para el VRPTW es:

$$x_{ij} = \begin{cases} 1, & \text{si se usa el arco}(i,j)\text{en una ruta} \\ 0, & \text{en caso contrario} \end{cases}$$

3.2.2 Variables auxiliares

u_i : Representan la carga acumulada que ha dejado un vehículo después de servir al cliente i. Son necesarias para las restricciones de eliminación de subtours.

v_i : Representa el instante de tiempo en que llega el vehículo al cliente i. Son necesarias para las restricciones de las ventanas de tiempo.

w_i : Representa el tiempo de espera del vehículo para comenzar el servicio del cliente i. Estas variables nos ayudan para que el inicio de servicio sea lo antes posible.

e_i : Representa el instante de tiempo en que inicia el servicio del cliente i.

Las siguientes variables se introducen para calcular el inicio de servicio del cliente j.

$$r_{ij} = \begin{cases} w_j, & \text{si } x_{ij} = 1 \\ 0, & \text{si } x_{ij} = 0. \end{cases}$$

Las variables y_i serán necesarias para garantizar el balanceo de la ruta.

$$y_i = \begin{cases} 1, & \text{si el vehículo llega antes del inicio de la venteana de tiempo del cliente}i \\ 0, & \text{en otro caso} \end{cases}$$

Finalmente, las siguientes variables son necesarias para calcular el balanceo de las rutas.

L_{min} : Duración de la ruta más corta.

L_{max} : Duración de la ruta más larga.

3.3 Objetivos del problema

En este trabajo consideraremos dos objetivos, uno económico y otro social.

1. Económico: Minimizar el costo total de ruteo,

$$min\ z_1 = \sum_{i \epsilon V} \sum_{j \epsilon V j \neq i} c_{ij} x_{ij}$$

2. Social: Balancear en tiempo las rutas:

$$min\ z_2 = L_{max} - L_{min}$$

3.4 Restricciones

Vamos a definir cada una de las restricciones necesarias para el modelo VRPTW.

1. **La cantidad de rutas que salen del depósito debe ser igual a la cantidad que llegan**

$$\sum_{\forall i \epsilon I} x_{0i} - \sum_{\forall i \epsilon I} x_{0i} = 0 \tag{3.1}$$

2. **A cada cliente j llega un solo arco, ya sea del almacén o de otro cliente**

$$\sum_{\forall i \epsilon V, i \neq j} x_{ij} = 1 \quad \forall j \epsilon I \tag{3.2}$$

3. **De cada cliente i sale un solo arco, ya sea hacia el almacén o hacia otro cliente**

$$\sum_{\forall j \epsilon V, i \neq j} x_{ij} = 1 \quad \forall j \epsilon I \tag{3.3}$$

4. **Capacidad de vehículos y eliminación de subtouers**

$$u_i - u_j + Qx_{ij} \leq Q - d_j, \quad \forall i, j \epsilon I, i \neq j \tag{3.4}$$

$$d_i \leq u_i \leq Q, \qquad \forall i \epsilon I \tag{3.5}$$

Obliga a que se formen caminos (abiertos) con clientes que la suma de sus demandas no supera el valor de Q.

5. **Restricciones para las ventanas de tiempo de los clientes**

Como una generalización de las restricciones de MTZ (Miller-Tucker-Zemlin)[1] de eliminación de subtours para el TSP se pueden obtener restricciones similares para el VRPTW [22]:

$$e_i - e_j + (b_i + s_i + t_{ij} - a_j)x_{ij} \leq b_i - a_j, \forall i, j \epsilon I, j \neq i.$$

Siguiendo las ideas similares a Desrochers y Laporte (1991) se pueden obtener las siguientes desigualdades válidas para el VRPTW:

$$e_i - e_j + (b_i + s_i + t_{ij} - a_j)x_{ij} + (b_i - s_j - t_{ji} - a_j)x_{ji} \leq b_i - a_j, \forall i, j \epsilon I, j \neq i.$$

Con estas restricciones se garantiza que si $x_{ij} = 1 \Rightarrow e_j = e_i + s_i + t_{ij}$, lo que significa que el inicio del servicio del cliente $j(e_j)$ es igual al inicio del servicio del cliente $i(e_i)$ más el tiempo de servicio del cliente $i(s_i)$, más el tiempo de trayecto entre los clientes i y $j(t_{ij})$.

Sin embargo, si analizamos estas expresiones, vemos que el lado izquierdo (e_j) es el tiempo de inicio de servicio del cliente j y por el lado derecho $e_i + s_i + t_{ij}$ es el tiempo de llegada al cliente j, por lo que estas igualdades sólo son ciertas cuando el vehículo llega dentro de la ventana de tiempo. Para eludir esta dificultad, se hace necesario introducir las variables auxiliares r_{ij} definidas anteriormente.

Note que,

$$r_{ij} = w_j x_{ij} \Leftrightarrow \left\{ r_{ij} \leq M x_{ij} w_j = \sum_{\forall i \epsilon V, i \neq j} r_{ij} \right.$$

Con estas variables podemos escribir

$$e_i - e_j + (b_i + s_i + t_{ij} - a_j)x_{ij} +$$
$$(b_i - s_j - t_{ji} - a_j)x_{ji} +$$
$$r_{ij} - r_{ji} \leq b_i - a_j, \qquad \forall i, j \epsilon I, j \neq i. \qquad (3.6)$$
$$a_i \leq e_i \leq b_i, \qquad \forall i \epsilon I \qquad (3.7)$$
$$e_i = v_i + \sum_{\forall j \epsilon V, j \neq i} r_{ij}, \qquad \forall i \epsilon I \qquad (3.8)$$
$$r_{ij} \leq M x_{ij}, \qquad \forall i \epsilon V, j \epsilon I \qquad (3.9)$$

Las restricciones (3.7) garantizan que el servicio de cada cliente se inicie dentro de su ventana de tiempo, las (3.8) relacionan el momento de inicio del servicio del cliente i con la llegada del vehículo y con el tiempo de espera, y las (3.9) obligan a las variables r_{ij} a tomar valores iguales a cero cuando no se utiliza el arco (i, j).

Cada par de valores de $i, j, j \neq i$, tienen asociado en el conjunto de restricciones (3.6) dos restricciones "simétricas":

$$e_i - e_j + (b_i + s_i + t_{ij} - a_j)x_{ij} + (b_i - s_j - t_{ji} - a_j)x_{ji} + r_{ij} - r_{ji} \leq b_i - a_j$$
$$e_j - e_i + (b_j + s_j + t_{ji} - a_i)x_{ji} + (b_j - s_i - t_{ij} - a_i)x_{ij} + r_{ji} - r_{ij} \leq b_j - a_i$$

Aquí existen tres casos posibles para las variables x_{ij} y x_{ji}:

a) Si $x_{ij} = x_{ji} = 0$ tenemos que, $e_i - e_j \leq b_i - a_j$ y $e_j - e_i \leq b_j - a_i$
Estas restricciones son redundantes, ya que por las restricciones (3.7), para la primer restricción $e_i \leq b_i$, $e_j \geq a_j$ y para la segunda restricción $e_j \leq b_j$, $e_i \geq a_i$.

b) Si $x_{ij} = 1 \Rightarrow x_{ji} = 0,\ r_{ji} = 0$, se obtiene que

$$\begin{cases} e_i - e_j + s_i + t_{ij} + r_{ij} \leq 0 \\ e_j - e_i - s_i - t_{ij} - r_{ij} \leq 0 \end{cases}$$

$$\Rightarrow \begin{cases} e_j \geq e_i + s_i + t_{ij} + r_{ij} \\ e_j \leq e_i + s_i + t_{ij} + r_{ij} \end{cases}$$

$$\Rightarrow e_j = e_i + s_i + t_{ij} + r_{ij}$$

Esto significa que si el cliente j es visitado inmediatamente después del cliente i, el tiempo de inicio del servicio del cliente j (e_j) es igual a la suma del tiempo de inicio de servicio del cliente i (e_i) más el tiempo de servicio del cliente i (s_i) más el tiempo en recorrer la distancia entre los clientes i y j (t_{ij}), más el tiempo de espera del vehículo para comenzar el servicio r_{ij}, si fuera necesario.

c) Si $x_{ji} = 1 \Rightarrow x_{ij} = 0,\ r_{ij} = 0$, y de forma similar al caso b), se obtiene que

$$\begin{cases} e_i - e_j - s_i - t_{ij} - r_{ij} \leq 0 \\ e_j - e_i + s_j + t_{ji} + r_{ji} \leq 0 \end{cases}$$

$$\Rightarrow \begin{cases} e_i \leq e_j + s_j + t_{ji} + r_{ji} \\ e_i \geq e_j + s_j + t_{ji} + r_{ji} \end{cases}$$

$$\Rightarrow e_i = e_j + s_j + t_{ji} + r_{ji}$$

Por otra parte, en relación al balanceo de las rutas, es necesarito evitar que las rutas se alarguen artificialmente, esto es, debe lograrse que el balanceo sea real. Para ello se introducen las variables y_i y las siguientes restricciones que obligan a iniciar el servicio lo antes posible:

$$v_i + My_i \leq a_i + M \qquad \forall i \epsilon I \qquad (3.10)$$

$$v_i + My_i \geq a_i \qquad \forall i \epsilon I \qquad (3.11)$$

Las restricciones (3.10) y (3.11) son equivalentes a $M(y_i - 1) \leq a_i - v_i \leq My_i$, las cuales obligan a las varibles y_i a ser iguales a 1 cuando el vehículo llega antes del inicio de la ventana de tiempo del cliente i y las obligan a ser iguales a cero cuando el vehículo llega dentro de la ventana de tiempo. Es decir, si el vehículo llega dentro de la ventana de tiempo ($v_i \geq a_i$), entonces, $a_i - v_i \leq 0$ y obliga a que $y_i = 0$ para que $-M \leq a_i - v_i \leq 0$. Si el camión llega antes ($v_i \leq a_i$), entonces $a_i - e_i \geq 0$ y obliga a que $y_i = 1$ para que $0 \leq a_i - v_i \leq M$.

$$\sum_{\substack{\forall j \epsilon V \\ j \neq i}} r_{ji} \leq My_i \quad \forall i \epsilon I \tag{3.12}$$

$$e_i + (b_i - a_i)y_i \leq b_i \quad \forall i \epsilon I \tag{3.13}$$

Las restricciones (3.12) obligan a que la suma $\sum_{\substack{\forall j \epsilon V \\ j \neq i}} r_{ji}$ sea cero cuando el vehículo llega dentro de la ventana de tiempo.

Por último, las restricciones (3.13), conjuntamente con las restricciones (3.6) y (3.7), obligan a que se inicie el servicio lo antes posible.

Ahora bien, al no considerar al depósito en las restricciones (3.6), no se está relacionando el valor de las variables e_i para los clientes que ocupan la primera posición en una ruta con el tiempo de tránsito entre el depósito y el primer cliente. Por ello, es necesario agregar las siguientes restricciones para que el servicio del primer cliente en una ruta comience lo antes posible:

$$v_i \geq t_{oi} x_{oi}, \quad \forall i \epsilon I \tag{3.14}$$

$$e_i \leq (t_{oi} - b_i)x_{oi} + r_{oi} + b_i, \quad \forall i \epsilon I \tag{3.15}$$

Si el cliente i es el primero en una ruta ($x_{oi} = 1$) y el vehículo llega después del inicio de su ventana de tiempo ($y_i = 0$), entonces $e_i = t_{oi}$ por (3.8), (3.14) y (3.15) . Si el cliente i es el primero en una ruta $x_{oi} = 1$ y el vehículo llega antes del inicio de su ventana de tiempo ($y_i = 1$), entonces $e_i = t_{oi} + r_{oi}$ pero, por (3.7) y (3.13), $e_i = a_i$. Para cualquier otro caso las restricciones (3.14) y (3.15) son redundantes.

Por último para relacionar el momento en que llega el vehículo a un cliente con el momento en que sale del cliente anterior en la ruta, son necesarias las siguientes restricciones:

$$e_i - v_j + (b_o + s_i + t_{ij})x_{ij} \leq b_o, \quad \forall i, j \epsilon I, i \neq j \tag{3.16}$$

Si $x_{ij} = 1 \Rightarrow v_j \geq e_i + s_i + t_{ij}$, es decir, que el momento en que llega el vehículo al cliente $j(v_j)$ debe ser mayor o igual a la suma del momento en que salió del ciente $i(e_i + s_i)$ mas el tiempo del trayecto entre i y $j(t_{ij})$.

6. **Restricciones para la ruta más larga:**

$$L_{max} \geq e_i + s_i + t_{io}x_{io} \tag{3.17}$$

Cuando el cliente i sea el último en una ruta, la varible $x_{io} = 1$, por lo que la ruta más larga será la que mayor valor tenga de $e_i + w_i + s_i + t_{io}x_{oi}$.

7. **Restricciones para la ruta más corta:**

$$L_{min} \leq e_i + s_i + t_{io}x_{io} + (1 - x_{io})b_o \tag{3.18}$$

El sumando $(1 - x_{io})b_o$ tiene la función de penalizar a los clientes que no sean los últimos en una ruta. Cuando el cliente i es el último en una ruta, $x_{io} = 1$ y $L_{min} \leq e_i + s_i + t_{io}$; y cuando no lo es, $x_{io} = 0$ y $L_{min} \leq e_i + s_i + b_o$. De aquí que L_{min} siempre será menor o igual que el valor de $e_i + s_i + t_{io}$ para la ruta más corta.

Por otra parte, el segundo objetivo de minimizar $z_2 = L_{max} - L_{min}$ obliga a que L_{max} tome el menor valor posible (la duración de la ruta más larga) y a que L_{min} tome el mayor valor posible (la duración de la ruta más corta).

De esta forma, el modelo matemático del problema bajo estudio es:

$$min(z_1, z_2) \text{ donde } z_1 = \sum_{i \epsilon V} \sum_{\substack{j \epsilon V \\ j \neq i}} c_{ij} x_{ij} \text{ y } z_2 = L_{max} - L_{min}$$

sujeto a:

$$\sum_{\forall i \epsilon I} x_{0i} - \sum_{\forall i \epsilon I} x_{0i} = 0 \tag{3.1}$$

$$\sum_{\forall i \epsilon V, i \neq j} x_{ij} = 1 \qquad \forall j \epsilon I \tag{3.2}$$

$$\sum_{\forall j \epsilon V, i \neq j} x_{ij} = 1 \qquad \forall j \epsilon I \tag{3.3}$$

$$u_i - u_j + Q x_{ij} \leq Q - d_j, \qquad \forall i, j \epsilon I, i \neq j \tag{3.4}$$

$$d_i \leq u_i \leq Q, \qquad \forall i \epsilon I \tag{3.5}$$

$$e_i - e_j + (b_i + s_i + t_{ij} - a_j) x_{ij} +$$

$$(b_i - s_j - t_{ji} - a_j) x_{ji} +$$

$$r_{ij} - r_{ji} \leq b_i - a_j, \qquad \forall i, j \epsilon I, j \neq i. \tag{3.6}$$

$$a_i \leq e_i \leq b_i, \qquad \forall i \epsilon I \tag{3.7}$$

$$e_i = v_i + \sum_{\forall j \epsilon V, j \neq i} r_{ij}, \qquad \forall i \epsilon I \tag{3.8}$$

$$r_{ij} \leq M x_{ij}, \qquad \forall i \epsilon V, j \epsilon I \tag{3.9}$$

$$v_i + M y_i \leq a_i + M \qquad \forall i \epsilon I \tag{3.10}$$

$$v_i + M y_i \geq a_i \qquad \forall i \epsilon I \tag{3.11}$$

$$\sum_{\substack{\forall j \epsilon V \\ j \neq i}} r_{ji} \leq M y_i \qquad \forall i \epsilon I \tag{3.12}$$

$$e_i + (b_i - a_i) y_i \leq b_i \qquad \forall i \epsilon I \tag{3.13}$$

$$v_i \geq t_{oi} x_{oi}, \qquad \forall i \epsilon I \tag{3.14}$$

$$e_i \leq (t_{oi} - b_i) x_{oi} + r_{oi} + b_i, \qquad \forall i \epsilon I \tag{3.15}$$

$$e_i - v_j + (b_o + s_i + t_{ij}) x_{ij} \leq b_o, \qquad \forall i, j \epsilon I, i \neq j \tag{3.16}$$

$$L_{max} \geq e_i + s_i + t_{io} x_{io} \tag{3.17}$$

$$L_{min} \leq e_i + s_i + t_{io} x_{io} + (1 - x_{io}) b_o \tag{3.18}$$

3.5 Solución exacta

Para determinar la frontera eficiente utilizamos el Método de las Ponderaciones. Primero se resuelve el modelo con cada uno de los objetivos de forma independiente. De ahí se obtienen los valores de z_1^{max} y z_2^{min}, cuando se resulve para el segundo objetivo. Con estos valores se normalizan los objetivos de la siguiente manera:

$$\tilde{z}_1 = \frac{z_1 - z_1^{min}}{z_1^{max} - z_1^{min}}$$

y

$$\tilde{z}_2 = \frac{z_2 - z_2^{min}}{z_2^{max} - z_2^{min}}$$

y se resuelven modelos con la misma región factible que el modelo bi-objetivo, pero con función objetivo $z = \alpha\tilde{z}_1 + (1-\alpha)\tilde{z}_2$, donde los valores de α varían en el intervalo $(0, 1)$.

Además, después de resolver el modelo para el objetivo 1, se agregan la restricción $L_{max} - L_{min} \geq 0$ para que la relajación lineal del modelo para el objetivo 2 no sea negativa. Después de resolver el modelo para el objetivo 2, se agrega la restricción $L_{max} - L_{min} \geq R_{max} - R_{min}$, donde R_{max} y R_{min} son los valores de las duraciones más larga y más corta respectivamente, que se obtuvieron de la solución óptima del modelo para el objetivo 2.

Cuando se resuleven los modelos con el objetivo $z = \alpha\tilde{z}_1 + (1-\alpha)\tilde{z}_2$ los valores de α se van incrementando de una resolución a otra y cada vez que se reulve uno de estos modelos, se actualiza la restricción $L_{max} - L_{min} \geq R_{max} - R_{min}$. Esto último ayuda a obtener mejores valores de las relajaciones lineales.

En el capítulo de experimentación computacional, se dedica una sección para mostrar los frentes de pareto óptimos encontrados de forma exacta para algunas instancias de Solomon de 25 clientes.

Capítulo 4

Metodologías propuestas para el Bi-ObjVRPTW

Como ya se ha mencionado, el problema bajo estudio es un problema bi-objetivo, donde los criterios a evaluar son la distancia y el otro el balance de tiempo, por lo que hablar de una solución óptima no es sencillo y se requiere un análisis para determinar qué alternativa o alternativas son las más adecuadas para ese problema. Este tipo de problemas con más de un criterio se conocen en la literatura como Programación Multi-objetivo (MOP, por sus siglas en inglés).

Entonces es necesario hablar sobre los problemas de Programación Multi-objetivo y los conceptos básicos, por lo que en la primera sección se abordarán conceptos importantes del MOP. En las siguientes secciones se describirán las metodologías que se proponen para el problema bajo estudio.

4.1 Conceptos básicos de la Programación Multi-objetivo

Si consideramos que todas las funciones objetivo se quieren minimizar, el problema de programación multi-objetivo sería:

$$\begin{aligned} &Minimizar && (f_1(x), f_2(x), \ldots, f_p(x)) \\ &s.a. && x \epsilon X \end{aligned}$$

donde:

- $x = (x_1, x_2, \ldots, x_n)$ son las variables de decisión.
- X es el conjunto de puntos factibles.
- f_i son cada uno de los objetivos o criterios.
- $f = (f_1, f_2, \ldots, f_p)$ es la función vectorial objetivo.
- $y = (y_1, y_2, \ldots, y_p)$ es una solución.
- $Y = f(X)$ es el espacio de objetivos o espacio imagen.

En este tipo de problemas no se tiene una solución óptima como en el caso de la Programación Mono-objetivo, pues ahora se tiene un vector de funciones objetivo, y al comparar dos vectores no es posible definir cuál de los dos vectores es menor o mayor que el otro, debido a que en $\Re^p$ $(p > 1)$ no existe un orden total.

Por lo tanto es necesario decidir cuando un vector es menor o mayor que otro, para ello se introduce un orden inspirado en el concepto de eficiencia desarrollado por Vilfredo Pareto en 1896 que se define a continuación:

Definición 4.1 *Una solución* $y = (y_1, y_2, \ldots, y_p)$ ***domina*** *(denotado* $\prec$*) a una solución* $z = (z_1, z_2, \ldots, z_p)$ *si y sólo si* $\forall\, i \,\epsilon\, \{1, \ldots, p\}$, $y_i \leq z_i$ $y \,\exists\, i \,\epsilon\, \{1, \ldots, p\}$, *tal que* $y_i < z_i$.

Definición 4.2 *Una solución* x^* *se dice que es* ***Pareto-óptima****,si y sólo si no existe otro vector* x *tal que* $z = f(x)$ *domine a* $y = f(x^*)$

En otras palabras, podemos decir que el punto x^* es un óptimo de Pareto si no existe un vector x que haga mejorar alguno de los objetivos sin que empeore de forma

simultánea alguno de los otros. Observamos que una solución en el sentido de Pareto al problema de optimización multi-objetivo no será única: la solución estará formada por el conjunto de todos los vectores no dominados, a los que se les conoce como el *Conjunto no dominado* o *Frente de Pareto*.

El orden de Pareto definido anteriormente es un orden parcial, por lo que en general las técnicas de Programación Multi-objetivo se centran en obtener el conjunto de puntos de Pareto o puntos eficientes, denotado de aquí en adelante como E, o bien una aproximación del mismo ($\widehat{E}$).

Cuando se resuelve un problema multi-objetivo, el método propuesto debería converger al frente de Pareto real.

4.2 Metodología I

La primera metodología que se propone para la solución al problema bi-objetivo de ruteo de vehículos con ventanas de tiempo, es un algoritmo basado en Búsqueda Dispersa (Scatter Search) y el proceso a seguir es el siguiente:

1. Construir una solución y mejorarla, esto se realiza hasta alcanzar el tamaño de población deseado $PopSize$.
2. Extraer de dicha población el conjunto $\widehat{E}$ de soluciones no dominadas, el cual es la primera aproximación al frente de Pareto.
3. Seleccionar/actualizar el conjunto de referencia *RefSet*, el cual contiene las p mejores soluciones y q soluciones diversas.
4. Combinar cada par de soluciones de $RefSet$ y aplicar el método de mejora a cada una de ellas.
5. Actualizar $\widehat{E}$.

6. Repetir del paso 3 al paso 5 hasta que el conjunto eficiente $\widehat{E}$ no tenga nuevas soluciones.

Un esquema del proceso anterior se observa en la Figura 4.1

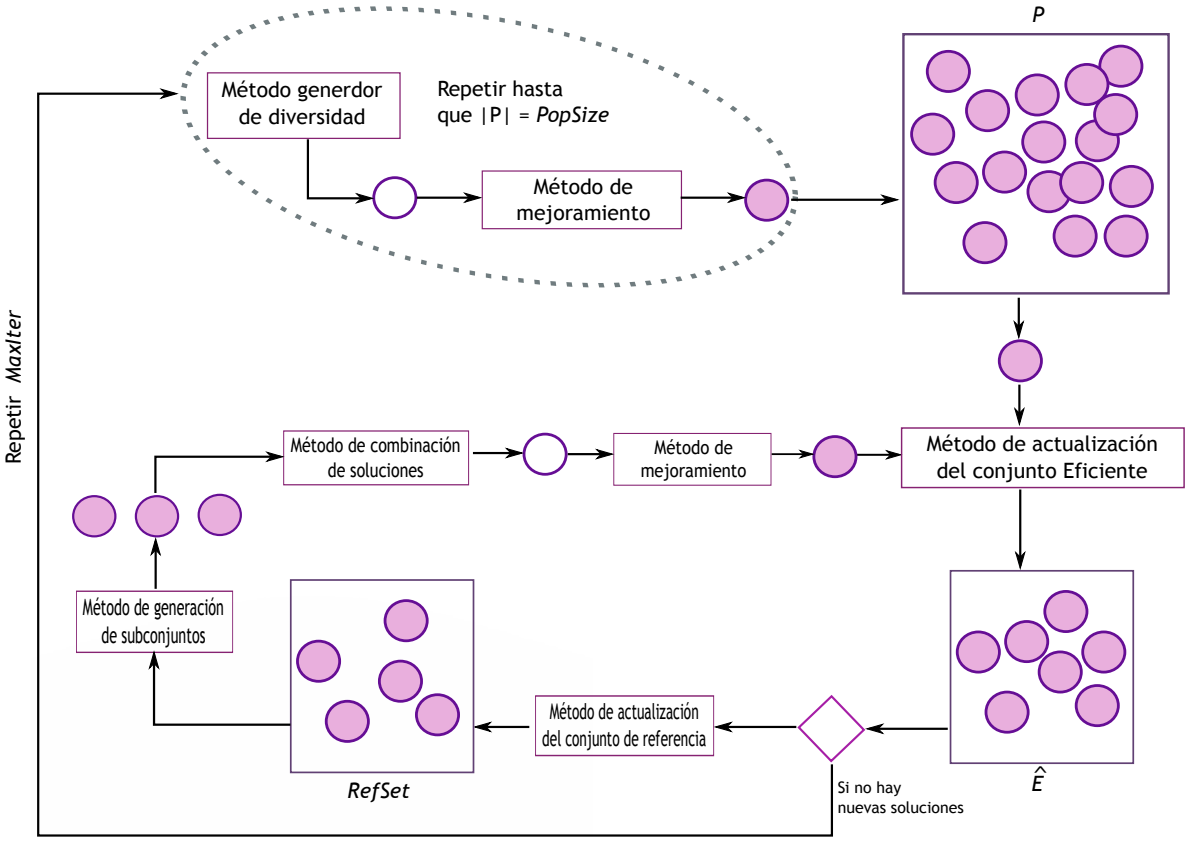


Figura 4.1: Proceso del Algoritmo

4.2.1 Construcción de la Población Inicial

A partir de las coordenadas (x_i, y_i) de los n clientes y del depósito (x_o, y_o) se debe calcular la matriz de distancia D, que será una matriz cuadrada de orden $n+1$, donde la distancia $d(i,j)$ entre dos puntos con coordenadas (x_i, y_i) y (x_j, y_j) se calcula usando la distancia euclideana $[(x_i - x_j)^2 + (y_i - y_j)^2]^{(1/2)}$.

Una vez que se ha calculado la matriz de distancia D, para construir una solución inicial se utiliza la idea propuesta por Beasley(1983)[2] que se describe

a continuación: primeramente se genera de forma aleatoria un tour dirigido que inicia y termina en el depósito, por ejemplo en la Figura 4.2 se muestra el tour $\{0, 1, 2, 3, 4, 5, 6, 7, 0\}$.

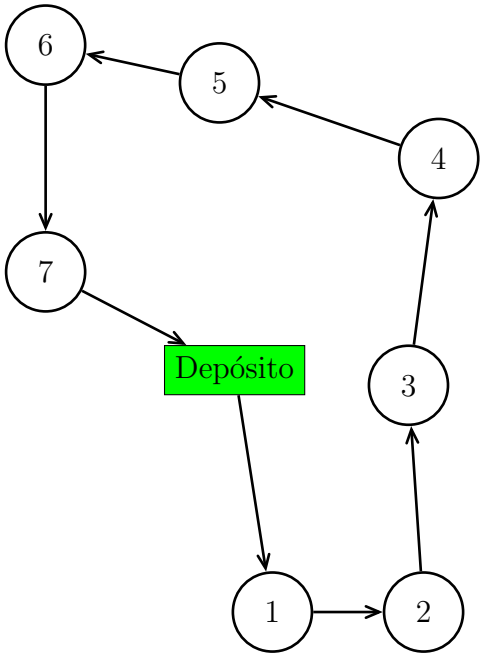


Figura 4.2: Tour Gigante

Una vez que se ha construido el tour, se aplica el operador 2-opt para mejorar si es posible el objetivo de la distancia. En la Figura 4.3 se ilustra el proceso del operador de cambio 2-opt: denotemos al depósito con un cuadrado y dado un tour, supongamos que elegimos el par de arcos $(i, i+1)$ y $(j, j+1)$ y que nos conviene reemplazarlos por los arcos (i, j) y $(i+1, j+1)$. Notemos que al realizar este movimiento tenemos que invertir la dirección de los clientes entre $i+1$ y j, como se muestra del lado derecho de la figura.

Una vez que se ha aplicado el operador 2-opt y no es posible mejorar más el objetivo de distancia utilizando este operador, el siguiente paso es construir, a partir de este tour gigante, una solución factible al problema de ruteo. Para ello se definirá un grafo auxiliar sobre el que se planteará un problema de ruta más corta. Los pesos de los arcos en este grafo auxiliar se definen de la siguiente manera (modificación de Beasley): c_{ij} es el costo total si es factible que el vehículo abastezca a los clientes $(i+1, i+2, \ldots, j, 0)(i < j)$, en otro caso tomará un valor muy grande.

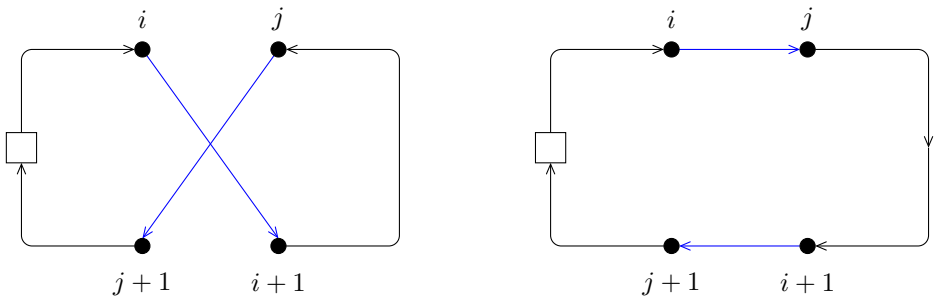


Figura 4.3: Operador de cambio 2-Opt

Formalmente definimos el costo como:

$$c_{ij} = \begin{cases} a_i + s_i + \sum_{k=i+1}^{j-1} r_k + d_{j0} & \text{si llega antes de inicio de la ventana de tiempo} \\ d_{0(i+1)} + \sum_{k=i+1}^{j-1} r_k + d_{j0} & \text{si llega dentro de la ventana de tiempo} \\ \infty & \text{en otro caso.} \end{cases}$$

donde

$$r_k = \begin{cases} a_k + s_k & \text{si llega antes de inicio de la ventana de tiempo} \\ d_{k(k+1)} + s_k & \text{si llega dentro de la ventana de tiempo} \end{cases}$$

Una vez calculados los pesos de los arcos en este grafo auxiliar, usamos el algoritmo de Dijkstra para encontrar la ruta más corta de 0 a n, si esta ruta contiene m arcos, entonces se necesitarán m vehículos para el problema de ruteo original.

Con el propósito de ilustrar, suponga que una vez aplicado el operador 2-opt al tour de la Figura 4.2 se obtuvo el tour $\{0, 5, 6, 7, 3, 4, 1, 2\}$. Entonces, si por ejemplo, la ruta más corta obtenida sobre el grafo auxiliar para este tour fuera $\{0, 7, 4, 2\}$ entonces una solución factible al problema original de ruteo sería la que se muestra en la Figura 4.4, en la que para satisfacer la demanda de 7 clientes enumerados del $\{1, \ldots, 7\}$ se necesitan 3 vehículos, donde cada ruta es una ruta factible en cuanto capacidad y ventanas de tiempo.

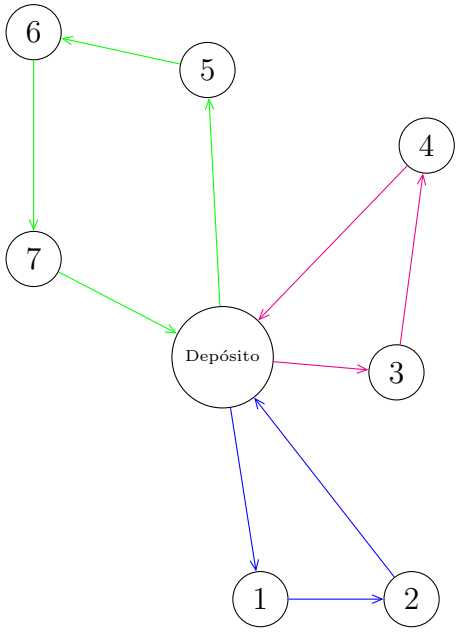


Figura 4.4: Construcción de 3 rutas factibles

4.2.2 Método de mejora

La idea de este método es usar un algoritmo de búsqueda local para mejorar las soluciones construidas para la población inicial y con el método de combinación.

El método de mejora es simple. Se toma una solución factible y se aplica una búsqueda local cuyo objetivo a optimizar es el balance de tiempo. El vecindario se define con el movimiento de reubicación, donde un cliente es removido de la posición actual de una ruta e insertado en otra ruta siempre que sea factible respecto a las ventanas de tiempo.

4.2.3 Construcción del conjunto eficiente

Inicialmente se construye una población de tamaño $PopSize$, sin embargo de este conjunto generado de soluciones factibles no todas son eficientes de acuerdo al criterio de Pareto, por lo tanto se debe extraer de allí el subconjunto de soluciones no dominadas, denotado como $\widehat{E}$.

4.2.4 Generación del conjunto de Referencia

Una vez que tenemos el conjunto eficiente $\widehat{E}$, el siguiente paso es formar el conjunto de Referencia el cual debe contener soluciones de buena calidad y diversas y se construye de la siguiente manera:

"p"soluciones de buena calidad, es decir, las mejores soluciones (no dominadas) para el objetivo de distancia y de balance de tiempo.

"q"soluciones diversas, las cuales se consideran usando el criterio max-min, es decir, aquellas soluciones de la población que maximizan la mínima de las distancias a las soluciones que ya están en el conjunto de referencia.

La distancia entre dos soluciones se calcula sumando el número de arcos no comunes [4].

4.2.5 Método de generación de subconjuntos

Este método genera subconjuntos de soluciones, los cuales serán usados para la generación de nuevas soluciones con el método de combinación. Existen diversas formas para la generación de subconjuntos [17], en este trabajo se formarán todos los subconjuntos posibles de tamaño dos.

4.2.6 Método de combinación de soluciones

Para la combinación de dos soluciones utilizaremos el operador "*order crossover*"denotado como OX [35]. Este será aplicado a las soluciones vistas como permutaciones o tours gigantes.

En la Figura 4.5 se muestra un ejemplo de la combinación OX. Supongamos que tenemos dos permutaciones $P1 = \{1, 3, 2, 6, 4, 5, 9, 7, 8\}$ y $P2 = \{3, 7, 8, 1, 4, 9, 2, 5, 6\}$ que son las soluciones que se van a combinar vistas como padres, y denotemos por

$H1$ y $H2$ las soluciones (permutaciones) resultantes de la combinación vistas como hijos. Hagamos dos cortes, en las posiciones i y j los cuales se generan de manera aleatoria, sean por ejemplo $i = 2$ y $j = 4$. Entonces de los padres tomamos la secuencia $P1(2), P1(3), P1(4)$ y $P2(2), P2(3), P2(4)$ las cuales se copian y se colocan en $H1(2), H1(3), H1(4)$ y $H2(2), H2(3), H2(4)$ respectivamente. Posteriormente se toman los clientes del segundo padre que aun no se hayan agregado a $H1$ hasta completar todos los elementos. Similarmente para el otro hijo $H2$.

			i		j				
$P1:$	1	3|	2	6|	4	5	9	7	8
$P2:$	3	7|	8	1|	4	9	2	5	6
$H1:$	8	1	2	6	4	9	5	3	7
$H2:$	2	6	8	1	4	5	9	7	3

Figura 4.5: Proceso de combinación de dos permutaciones

Una vez aplicando el proceso de combinación, observamos que obtenemos dos nuevas permutaciones, entonces procedemos a aplicar el algoritmo de Beasley para generar la solución asociada a cada permutación.

4.3 Metodología II

La segunda metodología que se propone es una adaptación de SS propuesta por Molina et al. [28], entre otras cosas, hacen una modificación al $RefSet$ para adaptarlo a las características especiales de la optimización multi-objetivo. Específicamente, las soluciones de calidad se miden considerando todas las funciones objetivos y las soluciones diversas se miden en el espacio de las funciones objetivo, mientras que en el caso mono-objetivo la diversidad se mide en el espacio de soluciones.

Cuando estamos en el caso de optimización multi-objetivo, queremos que el algoritmo propuesto sea capaz de obtener soluciones que cubran la frontera

eficiente, por tanto una medida que produce estos resultados deseados es considerar la diversidad en el espacio de las funciones objetivo.

A continuación se describen los componentes principales de nuestra segunda metodología de solución al Bi-OBjVRPTW, la cual toma ideas de MOAMP, descrito en el capítulo 2.

4.3.1 Descripción de los componentes de la implementación

En la Figura 4.1 se muestra un pseudo-código de la implementación:

Algoritmo 4.1: Pseudo-código de la metodología 2

```
begin
    CrearPoblacion(Pop,PopSize);
    GenerarConjuntoReferencia(RefSet, RefSetSize);
    repeat
        repeat
            SeleccionarSubconjunto(Subset, SubSetSize);
            CombinarSoluciones(SubSet, CurSol);
            MejorarSolucion(CurSol, ImpSol);
        until Criterio de parada 2;
        ActualizarConjuntoReferencia(RefSet);
    until Criterio de parada 1;
end
```

Ahora describamos cada uno de los componentes:

Método de CrearPoblacion

El procedimiento *CrearPoblacion* genera la población inicial. Para generar las soluciones iniciales, se utiliza el algoritmo de construcción de rutas en paralelo

propuesto por Potvin y Rousseau [34]. Este algoritmo va construyendo las rutas paralelamente, en lugar de secuencialmente.

El proceso para generar una solución inicial con num_routes rutas y de los clientes se van insertando en cualquiera de estas rutas. Una dificultad de esta estrategia en paralelo es decidir el número de rutas num_routes con las que se debe inicializar el proceso. En Potvin y Rousseau[34] aplican una sola vez el algoritmo secuencial de Solomon y con las rutas generadas, seleccionan como clientes semilla para cada ruta en paralelo al cliente que esté lo más alejado del depósito en cada una de estas rutas.

En nuestro caso, el enfoque será diferente al artículo ya mencionado. Lo primero que se realiza es calcular una cota inferior para el número de rutas:

$$LB_{num\_routes} = \left(\sum_{i=1}^{n} Demand(i) \right) / vehicle\_capacity$$

donde $Demand(i)$ es la demanda del cliente i, $i = 1, \ldots, n$. Entonces, para $j = LB_{num\_routes}, \ldots, UB$, donde UB es un parámetro que se define en el capítulo de experimentación computacional, se seleccionan de manera aleatoria j clientes como semilla para generar las rutas en paralelo. Posteriormente, ya que se han seleccionado los clientes semilla, el procedimiento continua como lo proponen Potvin y Rousseau. Si el número de semillas j no genera una solución factible para la instancia actual, entonces la solución parcial construida se descarta. El proceso se repite para un número de iteraciones max_iter dado, o hasta que se tenga el tamaño de población deseado $PopSize$.

Una vez que se ha construido una solución factible con j rutas, entonces a la solución actual se le aplica el procedimiento de mejora, el cual consiste en aplicar tres búsquedas locales en el orden siguiente:

```
1. LocalSearchDistance();
2. LocalSearchTimeBalance();
```

```
3. LocalSearchDistance();
4. for(k = 0; k < max_iter_commitment; k++)
LocalSearchCompromise();
```

En estas búsquedas locales el vecindario se define con el movimiento de reubicación, donde un cliente es removido de la posición actual de una ruta e insertado en otra ruta siempre que sea factible respecto a las ventanas de tiempo.

La función que guía a cada una de las búsquedas locales se define a continuación:

- *LocalSearchDistance*(): la función es la distancia total recorrida.
- *LocalSearchTimeBalance*(): esta búsqueda es guiada por la función de balance de tiempo de las rutas en la solución actual. Con el fin de evitar que el balance de las rutas sea artificial, cuando un cliente es reubicado en otra ruta el criterio a considerar es la posición en la que se tenga el menor incremento de la distancia total, siempre que exista más de una ubicación factible.
- *LocalSearchCompromise*(): para esta búsqueda local se utiliza la función que minimiza la distancia al punto ideal, donde este punto contiene los valores mínimos para cada objetivo i, en nuestro caso para $i = 1, 2$.

La idea de minimizar una función que mida la distancia al punto ideal, en *LocalSearchCompromise*() es para seguir la noción de la Programación Compromiso [36], donde suponemos que el centro decisor preferirá una solución que esté cercana al punto ideal sobre cualquiera que este más alejada.

Para calcular dicha distancia al punto ideal se utiliza la métrica L_∞ normalizada, pues en general, un punto que minimiza una distancia $L_p,\ p\epsilon[1,\infty]$, normalizada al punto ideal, es también un punto eficiente y tienen la característica común de representar un buen equilibrio entre los criterios, es decir, representan un buen valor para todos los objetivos. La fórmula para calcular la métrica L_∞ es la siguiente:

$$L_\infty(x) = max_{i=1,2}\left\{w_i\left(\frac{f(x) - f_i^{min}}{f_i^{max} - f_i^{min}}\right)\right\}$$

donde w_i son pesos aleatorios, f_i^{min} y f_i^{max} son los valores mínimo y máximo, respectivamente, para cada uno de los objetivos en el conjunto de soluciones construidas hasta el momento.

Posteriormente, una vez que se genera la población inicial del tamaño deseado $PopSize$, se genera el conjunto eficiente $EfficientSet$ el cual contiene puras soluciones no dominadas.

Método de GeneracionConjuntoReferencia

Recordemos que para el proceso de combinación la metodología del SS trabaja con un conjunto de menor tamaño denominado $RefSet$ el cual contiene soluciones del conjunto eficiente.

Sea b el tamaño del $RefSet$, el cual inicialmente se construye de la siguiente manera:

1. Seleccionar las r mejores soluciones del conjunto eficiente para cada uno de los objetivos.

2. Seleccionar las $b-2r$ soluciones de $EfficientSet \backslash RefSet$ tales que maximicen la distancia entre estos puntos y los que ya estan en el conjunto de referencia. Esta distancia se calcula de la siguiente manera:

$$L_\infty(x) = \sum_{i=1,2}\left(\frac{f_i(x) - f_i^{min}(RefSet \backslash Solution)}{f_i^{max} - f_i^{min}}\right)$$

donde f_i^{max} y f_i^{min}, son los valores máximo y mínimo de la *i-ésima* función objetivo, respectivamente, de $EfficientSet$.

Método de SeleccionarSubconjunto

Los subconjuntos se forman considerando todos los pares de elementos de $RefSet$. El siguiente paso es combinar las soluciones de cada uno de los subconjuntos.

Método de CombinarSoluciones

Sean S_1 y S_2 dos soluciones a combinar de los subconjuntos generados en el paso anterior. Denotemos por S' la nueva solución, la cual contendrá el menor número de rutas de las soluciones a combinar.

Para el proceso de combinación de dos soluciones, lo primero que se hace es agregarlas, si existen, las sub-rutas que son comunes en ambas soluciones. Después, se utiliza el algoritmo de construcción en paralelo de Potvin, para agregar los clientes que aún no están ruteados.

Método de MejorarSolucion

Para este proceso de mejorar las soluciones se utiliza la búsqueda local $LocalSearchCompromise$ descrita anteriormente, cuya función objetivo es la métrica L_∞, donde f^{max} y f^{min} se calculan considerando unicamente las soluciones S_1 y S_2.

Capítulo 5

Experimentación Computacional

Las metodologías propuestas en el capítulo anterior fueron implementadas computacionalmente, por lo que en este capítulo se presentan algunos resultados de los experimentos.

En la sección 1 de este capítulo se presenta la descripción de las instancias utilizadas en los experimentos. En la sección 2 se describen las métricas que se utilizarán para evaluar el rendimiento de nuestros algoritmos. En la sección 3 y 4, se describen los parámetros a ajustar y los resultados obtenidos para la metodología 1 y 2 respectivamente.

5.1 Descripción de las instancias

Para realizar la experimentación computacional de las metodologías propuestas, se han considerado las instancias de Solomon[38], diseñadas para los problemas de ruteo de vehículos con ventanas de tiempo (VRPTW). Las instancias varían de acuerdo a la ubicación geográfica de los clientes, capacidad de los vehículos y características de las ventanas de tiempo, tales como el tiempo asignado a cada cliente y el porcentaje de clientes con ventanas de tiempo muy ajustadas.

La información que se proporciona para cada cliente en estas instancias está en el orden siguiente: índice del cliente, ubicación geográfica con coordenadas x y y, demanda total, tiempo de inicio de servicio, tiempo final de servicio y tiempo

requerido para dar el servicio.

Se dispone de tres tamaños de instancias respecto al número de clientes: instancias con 25, 50 y con 100 clientes de pruba. Se considera que el tiempo de viaje entre los clientes es igual a la distancia euclídea correspondiente. Para cada tamaño hay 56 instancias que se dividen en seis categorías basadas en el patrón de la ubicación de los clientes y las ventanas de tiempo. Estas seis categorías son nombradas como C1, C2, R1, R2, RC1 y RC2.

Para las instancias de la categoría R, la ubicación geográfica de los clientes se determina de forma aleatoria, mientras que las instancias de la categoría C se refieren a clientes agrupados. O las instancias del tipo RC son una mezcla de clientes ubicados aleatoriamente y por grupos. Dado que el tiempo de viaje entre los clientes es igual a la distancia correspondiente, entonces, para los de la categoría R la distancia entre clientes es realtivamente grande comparada con los problemas de la categoría C.

Por otra parte, el conjunto de instancias está organizado respecto a las ventanas de tiempo. Los de la categoría 1 (R1, C1, RC1), tienen ventanas de tiempo estrechas (pequeñas) y los de la categoría 2 (R2, C2, RC2), han sido generados con ventanas de tiempo más grandes. Nótese que una ventana de tiempo muy pequeña propicia que muchas soluciones puedan ser infactibles por lo estrecho de las ventanas. Por el contrario, una ventana de tiempo más grande significa que es más probable encontrar más soluciones factibles, además favorece que se generen rutas más largas, es decir, que un vehículo pueda servir a un mayor número de clientes.

En la Figura 5.1 se grafican los seis tipos de instancias que podemos encontrar en Solomon, considerando 100 clientes. Principalmente se observa la generación aleatoria, por grupos o una combinación de las anteriores de la ubicación geográfica de los clientes.

Para evaluar el ajuste de los parámetros de los algoritmos, como el desempeño de las metodologías que se proponen, en la siguiente sección se describen las métricas que se consideraron.

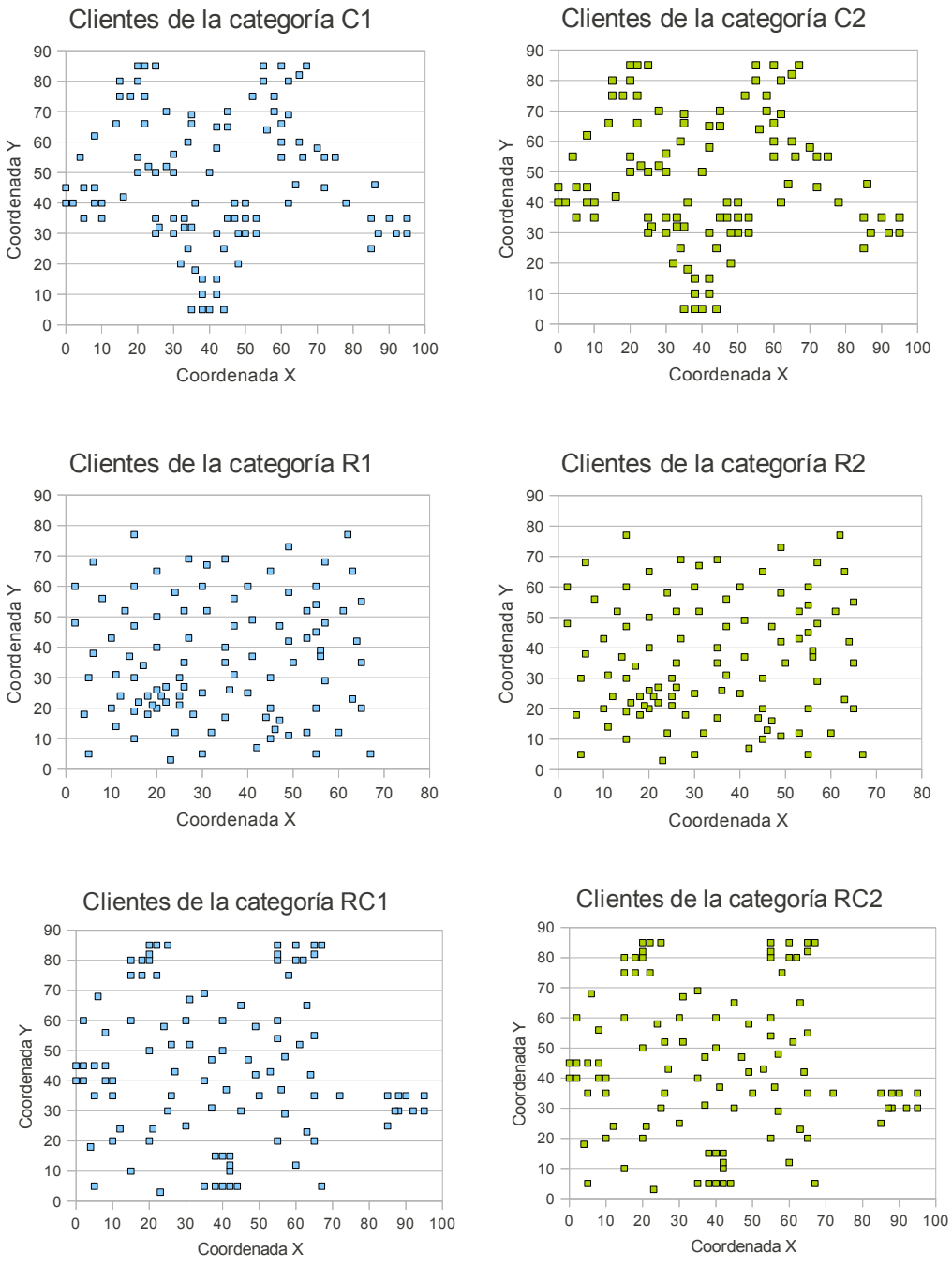


Figura 5.1: Grafica de los clientes para las categorías C1, C2, R1, R2, RC1 y RC2

5.2 Medidas de desempeño

Para evaluar el desempeño de las metodologías de solución para problemas de optimización multi-objetivo, se tienen algunos "instrumentos de medida". Para diseñar una buena métrica para este tipo de problemas se suelen tomar en cuenta tres cuestiones:

1. Minimizar la distancia del frente de Pareto producido por el algoritmo que se propone con respecto al verdadero frente de Pareto, suponiendo que lo conocemos.

2. Maximizar la distribución de las soluciones encontradas, de manera que se obtenga una distribución de soluciones no dominadas lo más "suave" y uniforme como sea posible.

3. Maximizar la cantidad de elementos del conjunto de Pareto encontrado.

Sin embargo, hasta ahora no existe una métrica que capture en un solo valor numérico los tres elementos anteriores e intentarlo sería inapropiado, pues cada uno de ellos se refiere a aspectos de desempeño distintos.

Por lo anterior, es recomendable considerar más de una métrica para evaluar el desempeño de un algoritmo. Nosotros utilizaremos las siguientes métricas:

1. Número de puntos en un frente.

2. Tamaño del espacio cubierto o SSC por sus siglas en inglés y como nos referiremos de aquí en adelante.

3. Medida de cobertura entre dos conjuntos, C(A,B) o también denotada como la métrica C.

Número de puntos en un frente

Esta métrica es fácil de implementar y se refiere a la capacidad del algoritmo de encontrar puntos eficientes. Es una medida importante, ya que las fronteras eficientes que proveen más alternativas para el centro decisor son preferibles que las fronteras con pocos puntos eficientes.

Tamaño del espacio cubierto

Zitzler y Thiele [43] propusieron la métrica denominada "Size of the Space Covered (SSC)". Esta métrica estima el tamaño del conjunto dominado global en el espacio de las funciones objetivo. La idea principal es calcular el área del espacio de las funciones objetivo cubierta por los vectores no dominados generados por el algoritmo de solución que se propone. La deficinción formal de esta medida se da a continuación:

Definición 5.1 *Sea* $X' = \{x_1, x_2, ..., x_k\} \subseteq X$ *el conjunto de los* k *vectores de decisión, donde* X *es el espacio de decisión. La función* $S(X')$ *da el volumen de la unión de los politopos* $p_1, p_2, ..., p_k$ *donde cada* p_i *está formado por la intersección de los siguientes hiperplanos derivados de* x_i*, junto con los ejes: para cada eje en el espacio objetivo, existe un hiperplano perpendicular al eje, el cual pasa por el punto* $(f_1(x_i), f_2(x_i), ..., f_n(x_i))$*. En el caso de dos dimensiones, cada* p_i *representa un rectángulo definido por los puntos* $(0, 0)$ *y* $(f_1(x_i),\ f_2(x_i))$*.*

En la Figura 5.2 se ilustra la métrica SSC.

Observemos que a mayor valor de SSC, la aproximación del frente de pareto es mejor.

Medida de cobertura entre dos conjuntos

Esta métrica fué propuesta por Zitzler y Thiele [43]. En esta medida compara dos conjuntos A y B de soluciones no dominadas, de forma que el valor $C(A, B)$ permite calcular mediante la ecuación 5.1 , el porcentaje de soluciones de B que son

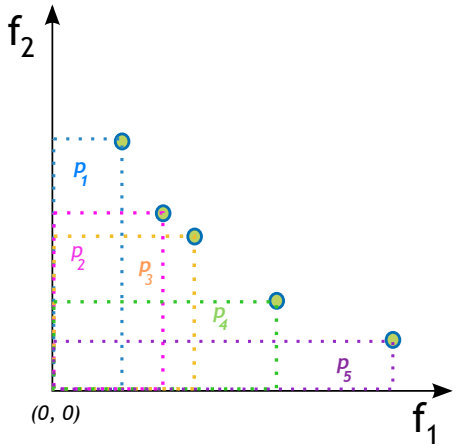


Figura 5.2: Gráfica en el espacio objetivo para calcular la métrica SSC

débilmente dominadas por al menos una solución de A.

$$C(A,B) = \frac{|b \in B | \exists a \in A, a \succeq b|}{|B|} \tag{5.1}$$

donde $|B|$ representa el número de soluciones contenidas en B. Observemos que cuando $C(A,B) = 1$, todas las soluciones de B son débilmente dominadas por las de A. Si $C(A,B) = 0$, ninguna solución de B es débilmente dominada por las de A.

Dado que $C(A,B)$ no es necesariamente igual a $1 - C(A,B)$, es necesario calcular siempre $C(A,B)$ y $C(B,A)$. Además, hay casos particulares en los que $C(A,B)$ no puede diferenciar los conjuntos A y B, como se muestra en la Figura 5.3.

5.3 Algunos Frentes de Pareto óptimos para instancias de Solomon

Recordemos que el problema que queremos resolver, como está planteado en el capítulo 3, consiste en minimizar dos objetivos: la distancia total recorrida y el

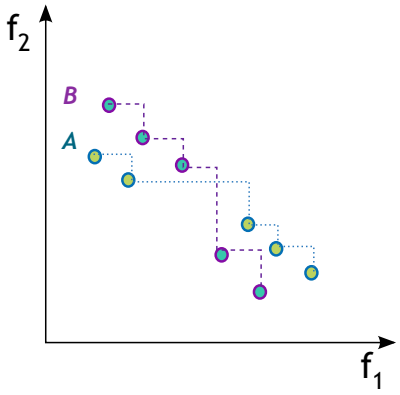


Figura 5.3: Problema en el uso de la métrica C: C(A,B) = C(B,A) = $\frac{3}{5}$

balanceo de las rutas, esto es

$$min\ z_1 = \sum_{i \epsilon V} \sum_{j \epsilon V j \neq i} c_{ij} x_{ij}$$

y

$$min\ z_2 = L_{max} - L_{min}.$$

Y que para determinar la frontera eficiente descrita en la sección 3.5 del capítulo 3 se utilizó el método de sumas ponderadas, donde la función objetivo para cada una de las combinaciones lineales esta dada por:

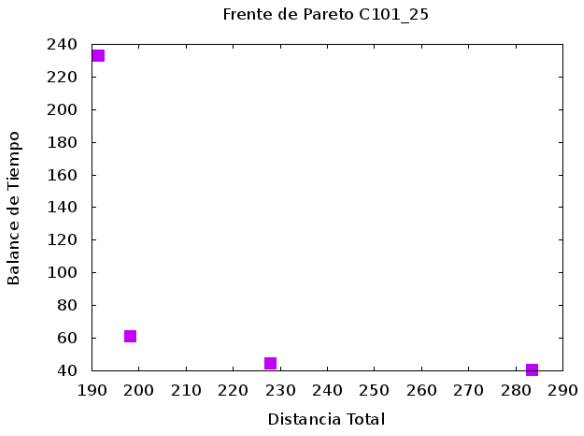
$$z = \alpha \tilde{z}_1 + (1 - \alpha)\tilde{z}_2,$$

y los valores de α varían en el intervalo $(0, 1)$. Entonces para cada instancia se están realizando 10 ejecuciones, esperando obtener a lo más 10 puntos de la frontera eficiente.

Las instancias de prueba se han ejecutado en una PC con procesador Pentium 4 con 3 GHz y 1 GB RAM, bajo Windows 2000. El código ha sido compilado usando Visual C++ 6.0. Las soluciones óptimas fueron encontradas con el solver comercial Cplex 11.1 (http://erdos.stanford.edu/cplex/).

En la Tabla 5.1 se muestran los valores obtenidos para la instancia C101 con 25 clientes, graficada del lado derecho de la misma, donde el eje de las coordenas x

representa el objetivo de la distancia total recorrida y el eje de las coordenadas y representa el objetivo del balance de tiempo de las rutas.



Obj_1	Obj_2
191.3	233.4
283.4	40.6
227.9	44.7
198.1	61.4

Tabla 5.1: Frente de Pareto para la instancia C101

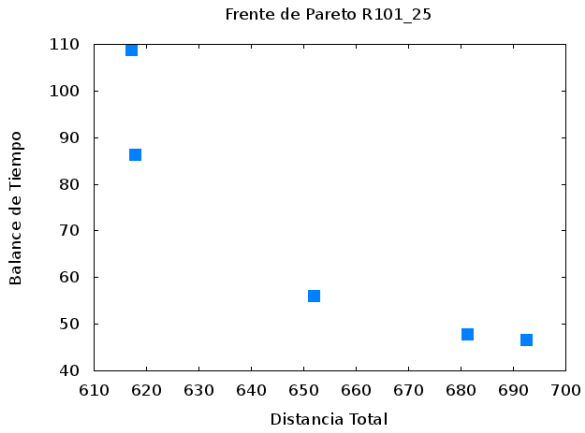
Otro ejemplo de los resultados obtenidos de forma exacta, se ilustra en la Tabla 5.2 para la instancia R101, en el que se obtienen cinco soluciones del frente de pareto.

Note que se obtienen pocos puntos del frente de pareto. En el Apéndice [**?**] se muestran los resultados obtenidos para las instancias de Solomon del tipo C1. Sin embargo debido, a la complejidad del problema, no fue posible en todos los casos encontrar los frentes de Pareto óptimos, por lo que se procedió a dar un límite de tiempo de dos horas a cada uno de los diez problemas que se resuelven para cada instancia.

5.4 Experimentos par evaluar la Metodología I

5.4.1 Ajuste de parámetros

Para la metodología I, los parámetros a ajustar considerados son:



Obj_1	Obj_2
617.1	108.7
692.5	46.7
681.2	47.9
651.9	56.1
617.8	86.3

Tabla 5.2: Frente de Pareto para la instancia R101

- $RefSEtSize$: El tamaño del conjunto de referencia, en el que se probaron los valores de: 8, 10 y 12.
- $PopSize$: El tamaño de la población inicial, el cual se tomará igual a 100 ó 200.

Para estudiar el comportamiento de los parámetros anteriores, estos fueron combinados obteniendo un total de 6 combinaciones con las que se resolvieron las siguientes instancias:

1. C101
2. C102
3. C201
4. C202
5. R101
6. R102
7. R201
8. R202
9. RC101
10. RC102
11. RC201
12. RC202.

Realizamos para cada instancia 10 repeticiones. Para que posteriormente se aplicaran las métricas ya descritas y tener para cada combinación la mejor iteración. Por ejemplo, para la instancia R101 en la Figura 5.4 se grafica la mejor iteración para cada combinación de parámetros.

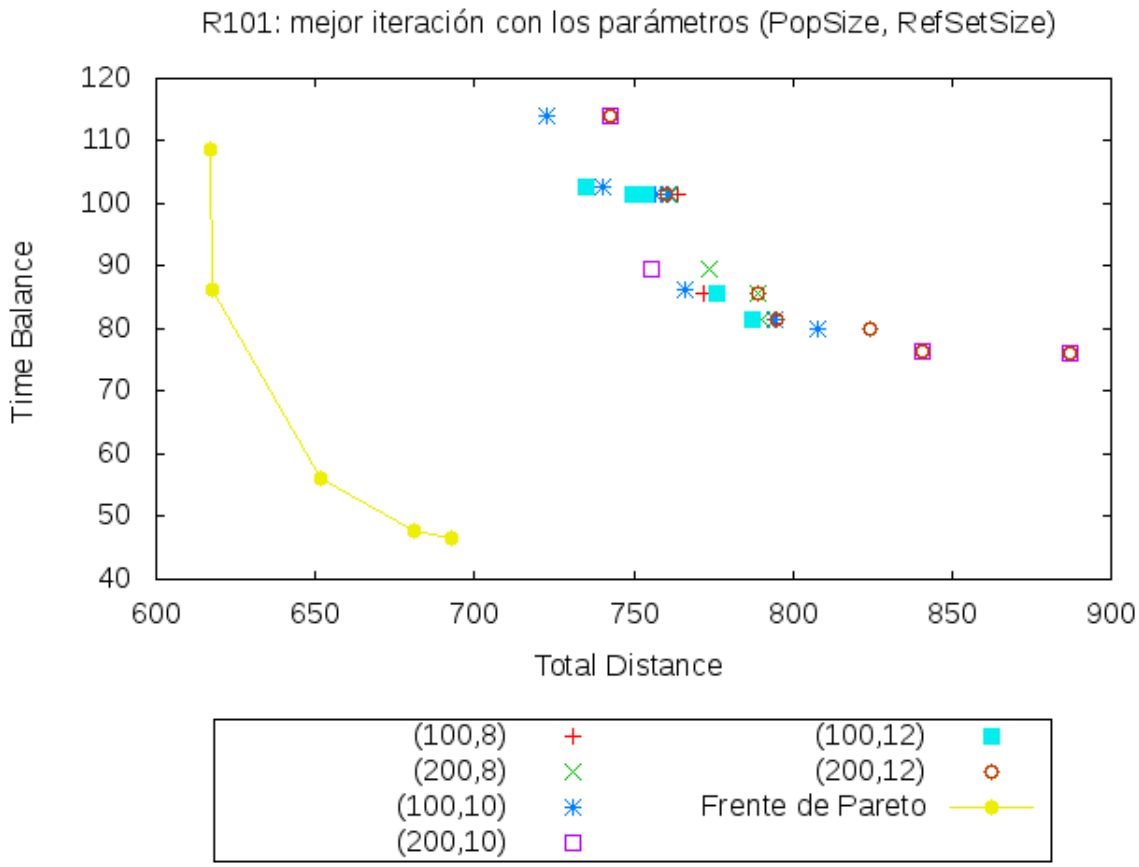


Figura 5.4: Comparando el FP óptimo con las aproximaciones obtenidas con la metodología I para R101

Observamos en la Tabla 5.4.1, que la mejor combinación de parámetros en promedio es cuando $PopSize = 200$ y $RefSetSize = 8$. Pues observemos que en promedio el mayor número de puntos obtenidos es al considerar el tamaño de la población de 200 y el conjunto de referencia de 10 elementos. Sin embargo, al observar el valor del SSC tenemos que el mayor valor lo obtenemos con la combinación de parámetros $(200, 8)$, y dado que la diferencia en promedio al comparar el número de puntos obtenidos con la combinación $(200, 10)$ no es grande, entonces le damos mayor peso al SSC.

Es siguiente proceso es comparar la aproximación obtenida para cada instancia obtenida al aplicar la metodología I contra la mejor aproximación que se obtenga

Combinación de Parámetros $(PopSize, RefSetSize)$	Número de Puntos	SSC
$(100, 8)$	6.167	0.654
	9.690	0.057
$(200, 8)$	6.417	0.742
	14.218	0.034
$(100, 10)$	6.333	0.702
	15.090	0.046
$(200, 10)$	6.667	0.659
	16.872	0.032
$(100, 12)$	5.667	0.681
	8.254	0.037
$(200, 12)$	6.667	0.658
	15.472	0.037

Tabla 5.3: En la primer fila se muestra el promedio y varianza (segunda fila) de las combinaciones de parámetros para la metodología I

aplicando la segunda metodología. Estos resultados se muestran en la última sección de este capítulo.

5.5 Experimentos para evaluar la Metodología II

5.5.1 Ajuste de parámetros

Las variaciones en el desempleño de la Metodología II están relacionadas a los valores de los siguientes parámetros:

- $PopSize$: Tamaño de la población inicial, el cual se tomará igual a 100 ó 200.
- $max\_n\_iter\_commitment$: Número de iteraciones para la búsqueda local compromiso. Se probaron los valores de 5 10 y 15 iteraciones.
- $(alpha1, alpha2)$: Son los pesos que se dan para la combinación lineal de

la función compromiso utilizada en el constructivo de Potvin. Las parejas consideradas son: $a = (0.25, 0.75), b = (0.5, 0.5)$ y $c = (0.75, 0.25)$.

Para estudiar el comportamiento de los parámetros anteriores, se combinaron sus posibles valores y con cada una de las 18 combinaciones resultantes se resolvieron las siguientes instancias:

1. C101
2. C102
3. C201
4. C202
5. R101
6. R102
7. R201
8. R202
9. RC101
10. RC102
11. RC201
12. RC202.

Para cada una de las instancias se realizaron 10 repeticiones. Para ejemplificar, en la Figura 5.5 se han graficado las 10 iteraciones, enumeradas de 0 a 9, para la instancia R101 con la combinación de parámetros: $PopSize = 100$, $alpha_1 = 0.25$, $alpha_2 = 0.75$, $max\_iter\_commitment = 5$. Ahora bien, debemos elegir la mejor de estas iteraciones, gráficamente no siempre resulta fácil, por ello es que aplicamos las métricas de comparación definidas anteriormente.

Siguiendo con la instancia R101, los valores arrogados por las métricas se observan en la Tabla 5.4, donde la primera columna aparece el número de la iteración, después aparece el Número de puntos y SSC.

Si elegimos la aproximación del frente de Pareto con el mayor número de puntos, de acuerdo a la tabla anterior, las aproximaciones a elegir serían la iteración 5 o 9, y si además consideramos el mayor valor de SSC tendríamos que la mejor opción es It_5, lo cual también se corrobora al considerar la métrica de cobertura, que se muestra en la Tabla 5.5:

Para cada una de las instancias de prueba, se aplicaron las mismas métricas para determinar la mejor de las 10 iteraciones. La información completa se encuentra en el Apéndice B.

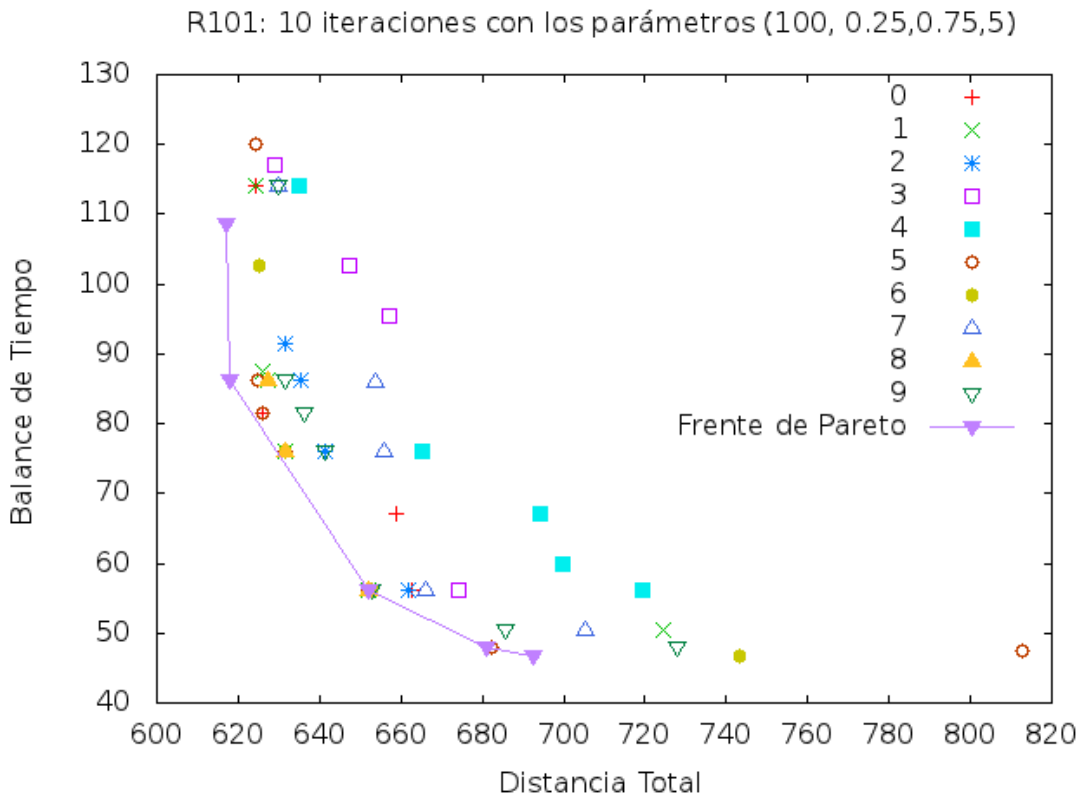


Figura 5.5: Ejemplo de la combinación de parámetros para la instancia R101

Siguiendo con nuestro ejemplo, en la la Figura 5.6 estamos graficando, para la instancia R101, el frente aproximado obtenido por la mejor iteración para cada combinación de parámetros versus el Frente de Pareto Óptimo (FPO) . Observamos que de forma gráfica no es sencillo decidir cuál de las aproximaciones es la "mejor".

Entonces, para determinar la mejor combinación de parámetros, volvemos a aplicar las métricas: SSC y número de puntos a cada instancia. Obteniendo que la mejor combinación de parámetros es: $PopSize = 200$, $alpha_1 = 0.5$, $alpha_2 = 0.5$, $max\_iter\_commitment = 10$. En la Figura 5.7 observamos la aproximación con los mejores parámetros para la instancia R101 y el frente de Pareto óptimo.

Con los parámetros ajustados, lo siguiente es establecer los componentes que serán aplicados finalmente en la metaheurística propuesta, antes de encontrar la mejor aproximación para cada uno de las instancias de prueba. En la siguiente sección

Instancia	NúmPtos	SSC
It_0	5	0.873058
It_1	6	0.900778
It_2	4	0.859537
It_3	4	0.768128
It_4	5	0.728159
It_5	7	0.929795
It_6	4	0.912771
It_7	5	0.82331
It_8	3	0.890819
It_9	7	0.894705

Tabla 5.4: Métricas para la instancia R101 y parámetros $PopSize = 100$, $alpha_1 = 0.25$, $alpha_2 = 0.75$, $max\_iter\_commitment = 5$

se mostrarán los resultados obtenidos.

5.5.2 Componentes ajustadas

Una vez definidos los parámetros, otro estudio importante que se realizó para la aplicación de la metodología a todas las instancias de prueba, fué definir las búsquedas que se iban a implementar para la mejora de las soluciones. Recordemos que las búsquedas definidas en la sección 4.3 del capítulo 4, son:

```
1. LocalSearchDistance();
2. LocalSearchTimeBalance();
3. LocalSearchDistance();
4. for(k = 0; k < max_iter_commitment; k++)
LocalSearchCompromise();
```

El análisis realizado consistió en determinar si la aplicación de la búsqueda compromiso y/o el proceso de combinación es estadísticamente significativa en la calidad de la aproximación del frente de Pareto obtenido.

Las combinaciones que se estudiaron en este análisis de componentes han sido:

$C(It_i, It_j)$	It_0	It_1	It_2	It_3	It_4	It_5	It_6	It_7	It_8	It_9
It_0	0	0.333333	0.75	1	1	0.142857	0	0.8	0.333333	0.571429
It_1	0.4	0	1	1	1	0.142857	0.25	0.8	0	0.714286
It_2	0.2	0	0	0.75	1	0	0	0.6	0	0
It_3	0	0	0	0	0.6	0	0	0	0	0
It_4	0	0	0	0	0	0	0	0	0	0
It_5	0.4	0.5	1	1	1	0	0.5	1	0.333333	1
It_6	0.4	0	1	1	1	0.142857	0	0.8	0	0.571429
It_7	0	0.166667	0	0.5	1	0	0	0	0	0
It_8	0.4	0	1	1	1	0	0.25	0.8	0	0.714286
It_9	0.4	0.166667	0.75	0.75	1	0	0	0.8	0	0

Tabla 5.5: Matriz de cobertura para la instancia R101

- LSC_C: Aplicar ambos procesos, búsqueda local compromiso y combinación.
- NLSC_C: Aplicar el proceso de combinación pero no la búsqueda compromiso.
- LSC_NC: Aplicar la búsqueda compromiso pero no la combinación.

donde, LSC es LocalSearchCompromise(); NLSC: No LocalSearchCompromise(); C: Combination y NC: No Combination.

Los resultados que obtuvimos al aplicar los procesos anteriores a las instancias de prueba con 25 clientes fueron:

	Núm de puntos	SSC
LSC_C	10.464	0.853
	34.326	0.012
LSC_NC	10.536	0.863
	35.999	0.012
NLSC_C	9.375	0.826
	24.566	0.016

Tabla 5.6: Comparación de los componentes para la metodología II. Se muestra el promedio y desviación estándar de cada uno.

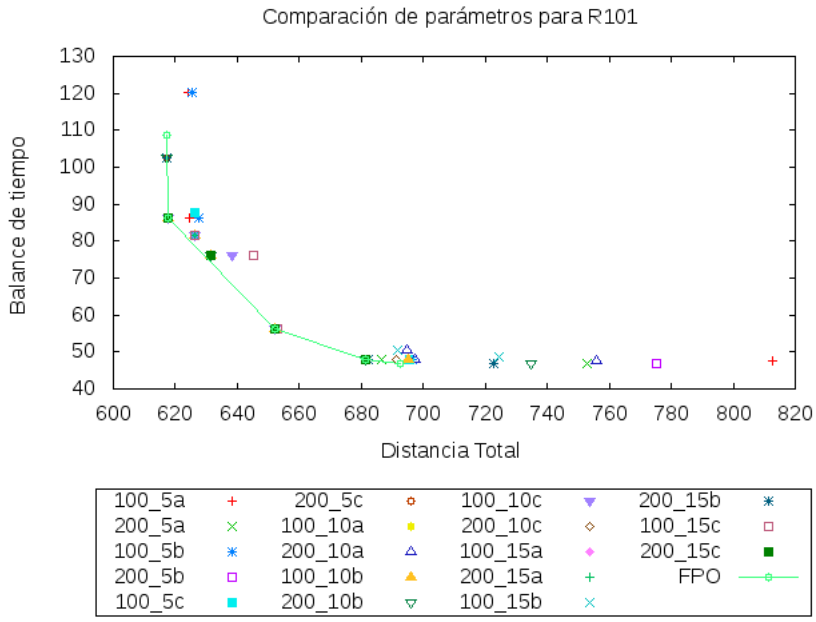


Figura 5.6: Comparando el FP óptimo con las aproximaciones obtenidas con la metodología II para R101

En la Tabla 5.6 se muestra el promedio (en la primera fila) y la desviación estándar (segunda fila) para las métrica: Número de puntos y SSC para las tres combinaciones aplicadas a las 56 instancias de Solomon de 25 clientes. Observamos que LSC_C y LSC_NC obtienen respecto a NLSC_C los mejores valores para ambos casos. A pesar de que las medidas anteriores muestran que es mejor aplicar combinación pero no aplicar la búsqueda local compromiso (NLSC_C), si observamos ambos valores no son muy diferentes a los obtenidos si aplicamos ambos componentes. veamos ahora que si aplicamos la métrica de Cobertura estas dos, obtenemos:

	LSC_C	NLSC_C
LSC_C	0	0.48
NLSC_C	0.13	0

Lo cual nos muestra que en promedio LSC_C tiende a dominar a los puntos eficientes encontrados por NLSC_C.

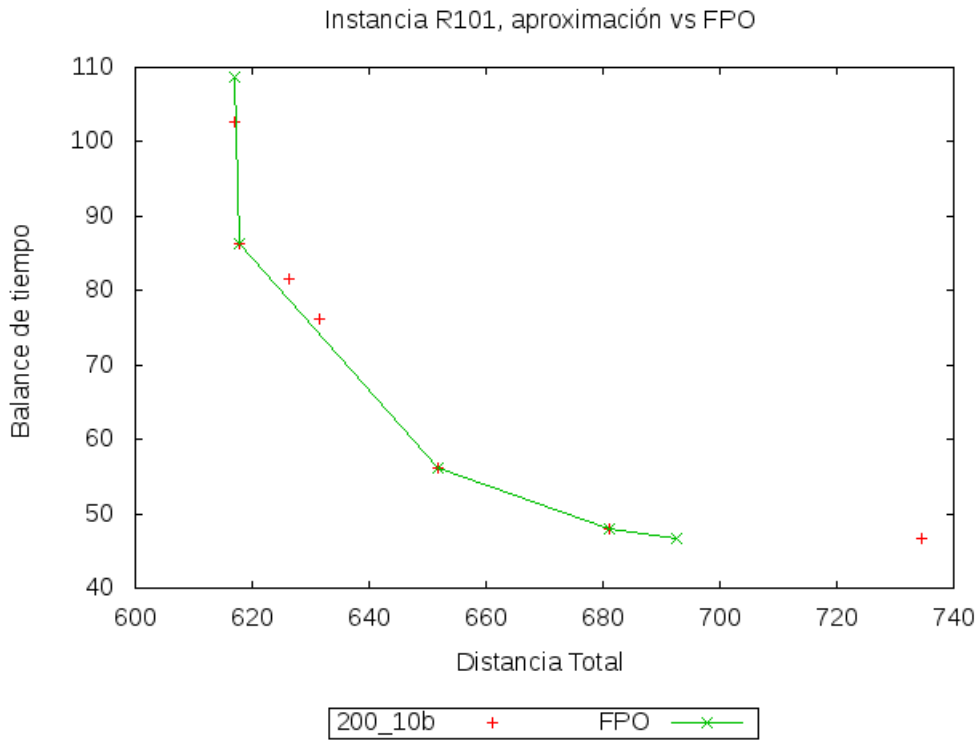


Figura 5.7: Comparando el Frente de Pareto Óptimo (FPO) con la aproximación obtenida con la metodología II, para la instancia R101.

Por lo tanto, podemos concluir que al aplicar la búsqueda local compromiso y la combinación obtenemos puntos eficientes que tienden a dominar a los puntos obtenidos si quitásemos alguno de los dos componentes, además que en promedio obtenemos casi el mismo número de puntos eficientes en ambos casos. Por consiguiente decidimos aplicar ambos componentes en la metodología.

Una vez definido los parámetros y definido los componentes que serán utilizados en la metodología, se resolvieron todas las instancias de prueba. En la siguiente subsección se mostrarán los resultados obtenidos.

5.5.3 Aproximaciones del Frente de Pareto

Resumiendo, los parámetros y componentes que serán aplicados a la metaheurística propuesta, son $PopSize = 200$, $alpha_1 = 0.5$, $alpha_2 = 0.5$, $max\_iter\_commitment = 10$ y $LocalSearchCompromise()$, $CombinarSoluciones$.

En la siguiente sección se comparan de forma gráfica alguno de los resultados obtenidos para las metodologías propuestas.

5.6 Comparación de metodologías

Ahora para comparar los resultados que se han obtenido en las instancias de prueba. En la Figura 5.8 se ejemplifica la comparación de la aproximación del frente de Pareto por cada una de las metodologías propuestas, observamos que sin recurrir a las métricas la metodología II obtiene mejores resultados.

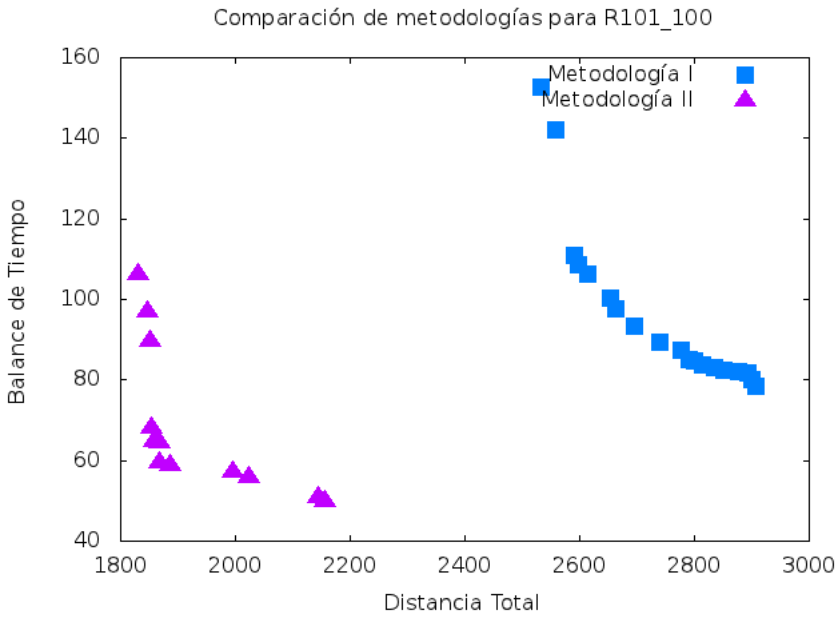


Figura 5.8: Comparando la aproximación obenida por cada metodología para la instancia R101

Lo anterior se observa todos los casos, por lo que basta el método gráfico para optar por la metodología II.

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones

En este trabajo se han propuesto dos metodologías de solución basadas en técnicas metaheurísticas. La primera está basada en la metodología de la Búsqueda Dispersa (SS) con una construcción inicial secuencial. La segunda metodología esta basada en una Búsqueda Dispersa Multiobjetivo, cuya construcción inicial se realiza en paralelo, se ha tomado la idea propuesta por Caballero y Molina [9], realizando búsquedas enlazadas, lo cual ha dado muy buenos resultados ha nuestro problema, ya que en instancias pequeñas podemos observar de forma gráfica que obtenemos puntos sobre el frente de Pareto Real.

Se realizaron diversos experimentos computacionales para combrobar la eficiencia y precisión de los algorimos propuestos al problema bi-objetivo bajo estudio.

Después de los resultados mostrados en el capítulo 5 y tomando encuenta toda la experimentación que se ha llevado a cabo para ambas metodologías, podemos concluir que la metodología II propuesta logra mejores resultados que la primera.

6.2 Trabajo a futuro

Ahora nos queda como trabajo a futuro, comparar las soluciones que actualmente lleva a cabo la empresa con soluciones que se obtengan al aplicar la metodología II, ya que por cuestión de tiempo ya no fué posible incluirlas en este trabajo.

Apéndice A

Frentes de Pareto Óptimo para algunas instancias de prueba

En este Apéndice se muestras los Frentes de Pareto Óptimos obtenidos al aplicar el modelo matemático propuesto en el capítulo 3.

En la Tabla A.1 se muestran los Frentes de Pareto obtenidos para las instancias de Solomón del tipo C1, el orden en que se muestra la información es: nombre de la instancia, valor de la función objetivo 1, valor alcanzado de la función objetivo 2 y finalmente el $GAP_{relativo}$.

	Obj_1	Obj_2	$GAP_{relativo}$
C101	191.3	233.4	0
	283.4	40.6	0
	227.9	44.7	0
	198.1	61.4	0
C102	190.3	387	0
	457.9	0.6	1
	255.8	0.6	0.5073
	236.8	4.1	0.341032
	235.6	4.8	5.1e-5
	225.2	14.5	6.02e-5
	222.1	17.8	0

	Obj_1	Obj_2	$GAP_{relativo}$
	202.5	46.6	0
	201.4	52.1	0
C103	190.3	387	0.093
	496.3	3.6	1
	235.6	0.5	3.3578
	234.8	0.7	2.025
	234.8	0.7	1.719
	210.6	24.1	1.66
	198	39.2	1.412
	194.9	65.9	2.39
C104	186.9	296.4	0.1877
	600	3.1	1
	243.7	0.2	4.877
	241.9	0.5	2.346
	239.1	0.5	2.051
	231.7	2	2.022
	194.1	38	1.99
C105	191.3	233.4	0
	511.9	0.4	1
	236.2	0.6	0
	199.3	24.8	0
C106	191.3	233.4	0.079
	298.8	27.6	0
	265.7	27.6	0
	198.1	64.9	0
	194.6	114.2	0
C107	191.3	233.4	0
	494.8	8.5	1

	Obj_1	Obj_2	$GAP_{relativo}$
	198.1	9.1	2.07
C108	191.3	233.4	0
	536.5	2.5	1
	286.1	0.1	1.09
	235.1	3	0.55
	201.4	9.3	0
	198.1	20.6	0
C109	191.3	233.4	0
	530	8	1
	240.5	0.3	1.17
	235.3	1	2.48
	227.8	7.4	1.42

Tabla A.1: Frente de Pareto para las instancias de Solomon del tipo C1

Además de estas, también se tiene la información del resto de las instancias, sin embargo no se muestran en este apéndice, pero esta información esta disponible.

Apéndice B

Definición de parámetros

En este Apéndice se muestra la mejor iteración de cada una de las instancias de prueba para la definición de los parámetros, lo cual se resume en la Tabla B.1, en la primer columna se muestra el nombre de la instancia (por ejemplo $C101\_5a$, indica que es la instancia de Solomon C101, $max\_iter\_commitment = 5$ y $(alpha_1, alpha_2) = (0.25, 0.75)$), la segunda columna nos indica la mejor iteración cuando $PopSize = 100$ y la tercer columna muestra la mejor iteración para $PopSize = 200$:

	Tamaño de la Población	
Instancia	100	200
$C101\_5a$	4	7
$C101\_5b$	0	0
$C101\_5c$	5	8
$C101\_10a$	6	6
$C101\_10b$	9	1
$C101\_10c$	8	0
$C101\_15a$	2	7
$C101\_15b$	1	7
$C101\_15c$	6	3
$C102\_5a$	3	3
$C102\_5b$	7	6
$C102\_5c$	7	7

	Tamaño de la Población	
Instancia	100	200
$C102\_10a$	5	8
$C102\_10b$	0	0
$C102\_10c$	5	3
$C102\_15a$	4	1
$C102\_15b$	1	1
$C102\_15c$	7	6
$C201\_5a$	1	1
$C201\_5b$	9	3
$C201\_5c$	0	4
$C201\_10a$	0	0
$C201\_10b$	2	0
$C201\_10c$	2	0
$C201\_15a$	8	0
$C201\_15b$	2	0
$C201\_15c$	0	0
$C202\_5a$	3	4
$C202\_5b$	0	0
$C202\_5c$	3	2
$C202\_10a$	0	3
$C202\_10b$	7	3
$C202\_10c$	9	7
$C202\_15a$	1	9
$C202\_15b$	3	8
$C202\_15c$	4	1
$R101\_5a$	5	2
$R101\_5b$	5	7
$R101\_5c$	7	4
$R101\_10a$	6	4

	Tamaño de la Población	
Instancia	100	200
$R101\_10b$	6	7
$R101\_10c$	4	2
$R101\_15a$	7	9
$R101\_15b$	0	7
$R101\_15c$	5	8
$R102\_5a$	9	6
$R102\_5b$	4	0
$R102\_5c$	9	7
$R102\_10a$	8	2
$R102\_10b$	6	5
$R102\_10c$	8	8
$R102\_15a$	4	7
$R102\_15b$	7	7
$R102\_15c$	0	0
$R201\_5a$	8	2
$R201\_5b$	6	6
$R201\_5c$	4	4
$R201\_10a$	5	5
$R201\_10b$	4	7
$R201\_10c$	1	6
$R201\_15a$	0	9
$R201\_15b$	1	3
$R201\_15c$	4	8
$R202\_5a$	5	2
$R202\_5b$	9	9
$R202\_5c$	2	1
$R202\_10a$	1	4
$R202\_10b$	0	6

	Tamaño de la Población	
Instancia	100	200
$R202\_10c$	3	0
$R202\_15a$	4	2
$R202\_15b$	1	5
$R202\_15c$	0	1
$RC101\_5a$	1	7
$RC101\_5b$	4	2
$RC101\_5c$	2	4
$RC101\_10a$	8	7
$RC101\_10b$	8	6
$RC101\_10c$	4	5
$RC101\_15a$	8	5
$RC101\_15b$	6	1
$RC101\_15c$	1	7
$RC102\_5a$	0	1
$RC102\_5b$	6	8
$RC102\_5c$	1	0
$RC102\_10a$	3	9
$RC102\_10b$	0	9
$RC102\_10c$	7	1
$RC102\_15a$	7	5
$RC102\_15b$	1	8
$RC102\_15c$	4	0
$RC201\_5a$	7	5
$RC201\_5b$	1	2
$RC201\_5c$	9	7
$RC201\_10a$	5	8
$RC201\_10b$	1	3
$RC201\_10c$	1	4

	Tamaño de la Población	
Instancia	100	200
$RC201\_15a$	0	0
$RC201\_15b$	8	6
$RC201\_15c$	7	3
$RC202\_5a$	3	8
$RC202\_5b$	1	2
$RC202\_5c$	9	3
$RC202\_10a$	6	6
$RC202\_10b$	4	9
$RC202\_10c$	1	9
$RC202\_15a$	5	5
$RC202\_15b$	2	6
$RC202\_15c$	2	4

Tabla B.1: Mejor iteración para cada combinación de parámetros

Bibliografía

[1] Ascheuer, N., M. Fischetti y M. Grötschel, «A polyhedral study of the asymmetric traveling salesman problem with time windows», *Networks*, **36**(2), págs. 69–79, 2000.

[2] Beasley, J., «Route first–Cluster second methods for vehicle routing», *Omega*, **11**(4), págs. 403–408, 1983.

[3] Beausoleil, R. P., «"MOSS"multiobjective scatter search applied to non-linear multiple criteria optimization», *European Journal of Operational Research*, **169**(2), págs. 426 – 449, 2006.

[4] Belfiore, P. y L. Fávero, «Scatter search for the fleet size and mix vehicle routing problem with time windows», *Central European Journal of Operations Research*, **15**, págs. 351–368, 2007.

[5] Borgulya, I., «An algorithm for the capacitated vehicle routing problem with route balancing», *Central European Journal of Operations Research*, **16**, págs. 331–343, 10.1007/s10100-008-0062-2, 2008.

[6] Braysy, O., W. Dullaert y M. Gendreau, «Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows», *Journal of Heuristics*, **10**, págs. 587–611, 10.1007/s10732-005-5431-6, 2004.

[7] Braysy, O. y M. Gendreau, «Vehicle Routing Problem with Time Windows, Part II: Metaheuristics», *Transportation Science*, **4**, págs. 119–139, 2005.

[8] BULLNHEIMER, B., R. F. HARTL y C. STRAUSS, «Applying the Ant System to the Vehicle Routing Problem», , 1997.

[9] CABALLERO R., R.-U. V., MOLINA J., «MOAMP: Programación Multiobjetivo mediante un procedimiento de Búsqueda Tabú», *Informe técnico*, Actas del II Congreso Español de Metaheurísticas y Algoritmos Evolutivos y Bioinspirados MAEB, 2003.

[10] CORBERÁN, A., E. FERNÁNDEZ, M. LAGUNA y R. MARTÍ, «Heuristic Solutions to the Problem of Routing School Buses with Multiple Objectives», *The Journal of the Operational Research Society*, **53**(4), págs. 427–435, Abril 2002.

[11] CORDEAU, J.-F. y G. LAPORTE, «Tabu Search Heuristics for the Vehicle Routing Problem», en R. Sharda, S. VoB, C. Rego y B. Alidaee (editores), *Metaheuristic Optimization via Memory and Evolution*, *Operations Research/Computer Science Interfaces Series*, tomo 30, Springer US, págs. 145–163, 10.1007/0-387-23667-8-6, 2005.

[12] DANTZIG, G. y J. RAMSER, «The Truck Dispatching Problem», *Management Science*, **6**(1), págs. 80–91, Octubre 1959.

[13] DEB, K., A. PRATAP, S. AGARWAL y T. MEYARIVAN, «A fast and elitist multiobjective genetic algorithm: NSGA-II», *Evolutionary Computation, IEEE Transactions on*, **6**(2), págs. 182 –197, abril 2002.

[14] DOERNER, K., A. FOCKE y W. J. GUTJAHR, «Multicriteria Tour Planning for Mobile Healthcare Facilities in a Developing Country», *Informe técnico*, 2005.

[15] FISHER, M. L. y JAIKUMAR, «A generalized assignment heuristic for vehicle routing», *Networks*, **11**, págs. 109–124, 1981.

[16] GARCIA-NAJERA, A. y J. BULLINARIA, «Bi-objective Optimization for the Vehicle Routing Problem with Time Windows: Using Route Similarity to Enhance Performance», en M. Ehrgott, C. Fonseca, X. Gandibleux, J.-K. Hao y

M. Sevaux (editores), *Evolutionary Multi-Criterion Optimization*, *Lecture Notes in Computer Science*, tomo 5467, Springer Berlin / Heidelberg, págs. 275–289, 10.1007/978-3-642-01020-0-24, 2009.

[17] GLOVER, F., M. LAGUNA y R. MARTÍ, «Scatter search», en *A. Ghosh, S. Tsutsui(Eds.), Advances in Evolutionary Computing: Theory and Aollications*, Springer, 2003.

[18] GLOVER, F. y L. M., *Tabu Search*, Kluwer Academic Publishers, 1997.

[19] HONG, S.-C. y Y.-B. PARK, «A heuristic for bi-objective vehicle routing with time window constraints», *International Journal of Production Economics*, **62**(3), págs. 249 – 258, 1999.

[20] JOZEFOWIEZ, N., F. SEMET y E.-G. TALBI, «Target aiming Pareto search and its application to the vehicle routing problem with route balancing», *Journal of Heuristics*, **13**, págs. 455–469, 10.1007/s10732-007-9022-6, 2007.

[21] JOZEFOWIEZ, N., F. SEMET y E.-G. TALBI, «Multi-objective vehicle routing problems», *European Journal of Operational Research*, **189**(2), págs. 293 – 309, 2008.

[22] KALLEHAUGE, B., J. LARSEN, O. B. MADSEN y M. M. SOLOMON, «Vehicle Routing Problem with Time Windows», en G. Desaulniers, J. Desrosiers y M. M. Solomon (editores), *Column Generation*, Springer US, págs. 67–98, 10.1007/0-387-25486-2_3, 2005.

[23] KNOWLES, J. y D. CORNE, «The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation», en *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, tomo 1, págs. 3 vol. (xxxvii+2348), 1999.

[24] LAPORTE, G., «The vehicle routing problem: An overview of exact and approximate algorithms», *European Journal of Operational Research*, **59**(3), págs. 345 – 358, 1992.

[25] LAPORTE, G. y F. SEMET, «Classical heuristics for the capacitated VRP», en *The vehicle routing problems. Toth, P and Vigo, D (eds)*, Philadelphia:SIAM, págs. 109–128, 2002.

[26] LIONG, C.-Y., I. WAN-ROSMANIRA, O. KHAIRUDDIN y M. ZIROUR, «Vehicle routing problem: models and solutions», *Journal of Qality Measurement and Analysis*, **4**(1), págs. 205–218, 2008.

[27] MOLE, R. H. y S. R. JAMESON, «A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion», *Operational Research Quarterly*, **27**(2), págs. 503–511, 1976.

[28] MOLINA, J., M. LAGUNA, R. MARTÍ y R. CABALLERO, «SSPMO: A scatter tabu search procedure for non-linear multiobjective optimization», *INFORMS Journal on Computing*, 2007.

[29] MONTANÉ, F. A. T. y R. D. G. AO, «A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service», *Computers and Operations Research*, **33**(3), págs. 595 – 619, 2006.

[30] MURATA, T. y R. ITAI, «Multi-objective Vehicle Routing Problems Using Two-Fold EMO Algorithms to Enhance Solution Similarity on Non-dominated Solutions», en C. A. Coello Coello, A. HernÃ¡ndez Aguirre y E. Zitzler (editores), *Evolutionary Multi-Criterion Optimization*, *Lecture Notes in Computer Science*, tomo 3410, Springer Berlin / Heidelberg, págs. 885–896, 10.1007/978-3-540-31880-4-61, 2005.

[31] NEBRO, A. J., F. LUNA, E. ALBA, A. B. B y B. DORRONSORO, «AbYSS: Adapting Scatter Search for Multiobjective Optimization», *Informe técnico*, IEEE Transactions on Evolutionary Computation, 2006.

[32] OMBUKI, B., B. ROSS y F. HANSHAR, «Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows», *Applied Intelligence*, **24**, págs. 17–30, 10.1007/s10489-006-6926-z, 2006.

[33] PACHECO, J. y R. MARTI, «Tabu search for a multi-objective routing problem», *Journal of Operation Research Society*, **57**, págs. 29–37, 2006.

[34] POTVIN, J.-Y. y J.-M. ROUSSEAU, «A parallel route building algorithm for the vehicle routing and scheduling problem with time windows», *European Journal of Operational Research*, **66**(3), págs. 331 – 340, 1993.

[35] PRINS, C., «A simple and effective evolutionary algorithm for the vehicle routing problem», *Computers and Operations Research*, **31**(12), págs. 1985–2002, 2004.

[36] ROMERO, C., *Teoría de la decisión multicriterio: Conceptos, técnicas y aplicaciones.*, Alianza Universidad-Textos, Madrid, 1993.

[37] RUSSELL, R. A. y W.-C. CHIANG, «Scatter search for the vehicle routing problem with time windows», *European Journal of Operational Research*, **169**(2), págs. 606 – 622, feature Cluster on Scatter Search Methods for Optimization, 2006.

[38] SOLOMON, M. M., «Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints», *Operations Research*, **35**(2), págs. 254 – 265, 1987.

[39] TAILLARD, E., P. BADEAU, M. GENDREAU, F. GUERTIN y J. POTVIN, «A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows», *Transportation Science*, **31**(2), págs. 170–186, Mayo 1997.

[40] TOTH, P. y D. VIGO, *The vehicle routing problem*, SIAM, 2002.

[41] ZENG, L., H. ONG y K. NG, «An assignment-based local search method for solving vehicle routing problems», *Asia-Pacific Journal of Operational Research*, **22**(1), págs. 85–104, 2005.

[42] ZITZLER, E., M. LAUMANNS y L. THIELE, «SPEA2: Improving the Strength Pareto Evolutionary Algorithm», *Informe Técnico 103*, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[43] Zitzler, E. y L. Thiele, «Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach», *Evolutionary Computation, IEEE Transactions on*, **3**(4), págs. 257 –271, nov 1999.

Ficha autobiográfica

Yadira Alondra De Santiago Badillo

Candidato para el grado de Maestría en Ciencias
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Un problema bi-objetivo de ruteo de vehículos con ventanas de tiempo

Nací en el municipio de Tolimán, Querétaro. La primaria y secundaria estudié en la "Abraham Lincoln", en Playas de Rosarito, Baja California, posteriormente regresé a Querétaro a continuar con mis estudios. Soy orgullosamente egresada de la Universidad Autónoma de Querétaro con Lic. en Matemáticas Aplicadas, en donde en mis últimos años de la carrera decidí seguir creciendo profesionalmente, con rumbo a la investigación de operaciones. Entonces terminé aquí, en el Posgrado de Ingeniería de Sistemas, conociendo más sobre la aplicación de las matemáticas a muchas áreas, que si bien ya me habían mencionado más de una vez, hasta ahora puedo ver la variedad de aplicaciones que existen.