# UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
## FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA


UN ENFOQUE BIOBJETIVO AL PROBLEMA DEL REPARADOR

POR

NANCY ARACELY ARELLANO ARRIAGA

EN OPCIÓN AL GRADO DE
MAESTRÍA EN CIENCIAS EN INGENIERÍA DE SISTEMAS

MAYO 2014

# UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
## FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
## SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO


UN ENFOQUE BIOBJETIVO AL PROBLEMA DEL REPARADOR

POR

NANCY ARACELY ARELLANO ARRIAGA

EN OPCIÓN AL GRADO DE
MAESTRÍA EN CIENCIAS EN INGENIERÍA DE SISTEMAS

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN, MAYO 2014

**Universidad Autónoma de Nuevo León**

**Facultad de Ingeniería Mecánica y Eléctrica**

**División de Estudios de Posgrado**

Los miembros del Comité de Tesis recomendamos que la Tesis «Un enfoque biobjetivo al problema del reparador.», realizada por el alumno Nancy Aracely Arellano Arriaga, con número de matrícula 1613608, 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

Dra. Irma Delia García Calvillo

Revisor

Dra. Iris Abril Martínez Salazar

Revisor

Vo. Bo.

Dr. Simón Martínez Martínez

División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, Mayo 2014

**Universidad Autónoma de Nuevo León**

**Facultad de Ingeniería Mecánica y Eléctrica**

**División de Estudios de Posgrado**

Los miembros del Comité de Tesis recomendamos que la Tesis «Un enfoque biobjetivo al problema del reparador.», realizada por el alumno Nancy Aracely Arellano Arriaga, con número de matrícula 1613608, 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

Dra. Irma Delia García Calvillo

Revisor

Dra. Iris Abril Martínez Salazar

Revisor

Vo. Bo.

Dr. Simón Martínez Martínez

División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, Mayo 2014

*A mis padres y a mi hermana.*

# Índice general

# Índice de figuras

# Índice de tablas

# Agradecimientos

Agradezco inicialmente a la Universidad Autónoma de Nuevo León por haberme abierto las puertas. A la Facultad de Ingeniería Mecánica y Eléctrica por el apoyo económico que me brindó durante toda la maestría, agradezco a CONACyT por haberme brindado una beca de manutención como estudiante de tiempo completo y a los doctores del programa de Posgrado en Ingeniería de Sistemas por todos los conocimientos compartidos.

Principalmente, le agradezco a la Dra. Ada Margarita Álvarez Socarrás por todo su apoyo tanto dentro como fuera de PISIS, le agradezco por su comprensión y paciencia infinita en todo lo relacionado a este trabajo. Por sus regaños ya que sin ellos nunca aprendería en que estoy mal ni podría mejorar, por su comprensión cuando me ausenté más de un mes y su apoyo en esos días y durante todo este tiempo realmente. Gracias por aguantarme tanto, con todas mis informalidades y distracciones.

A la Dra. Iris Abril Martínez Salazar, por todo su apoyo en el desarrollo de este trabajo. Gracias por escuchar mis traumas con c++, por ayudarme siempre que iba con dudas y por siempre haber estado disponible. Gracias Iris.

A la Dra. Irma Delia García Calvillo, le agradezco la confianza que ha puesto en mí desde la licenciatura, por su apoyo constante y sus palabras de aliento. Muchas gracias por seguir siempre en la mejor disposición y darme la confianza de poder recurrir a usted en cualquier momento.

Le agradezco al grupo de investigación de la Dra. Ada Álvarez y el Dr. Francisco Angel Bello Acosta, por escucharme en las reuniones y por sus acertados comentarios siempre. En especial, le agradezco a la M.C. Yadira Alondra De Santiago Badillo por todo su apoyo, gracias por ayudarme aunque no hayamos tenido el gusto de conocernos personalmente.

A mis compañeros de Yalma, sobre todo a los M.C. Fernando Elizalde, Cristina Maya, Nancy Arratia y Nelly Hernández, a los licenciados Juan Banda y Dulce Valdez así como los ingenieros Brenda Peña y Luis Benavides, porque son los que siempre están ahí para aguantarme y hacerme reír. Gracias por hacer de Yalma el laboratorio más divertido de todos. A mis compañeros de generación, sobre todo a los licenciados Juan Banda, Dory Álvarez, Lilian Llanas así como a los ingenieros Brenda Peña, Luis Benavides y Dago Quevedo por ser mis amigos.

A Adrián Mancilla Rivas, por su apoyo durante estos años. Gracias por aguantar las malas y las peores y seguir ahí.

Gracias especialmente a mis padres, por apoyarme desde el principio de mi vida. Gracias por su confianza cuando decidí venirme a esta ciudad a seguir estudiando, por estar siempre en todos los aspectos posibles. Sin ustedes y su apoyo, definitivamente nada de esto sería posible.Gracias a mi hermana, Lizeth Arellano por regañarme cuando lo necesito y hacerme reir y olvidar las penas cuando la veo. Veinte años contigo y un millón más, doy gracias por tenerte y haber aprendido que me haces más falta de lo que creí jamás. Te quiero enis.

Gracias a Babu (Mía) y a Neyma por hacer mis días mejores, así como a Bruno, Pocky, Norman y Nina por hacer de mis vacaciones toda una experiencia.

A todos, gracias.

Nancy Aracely Arellano Arriaga.

# Resumen

Nancy Aracely Arellano Arriaga.

Candidata para el grado de Maestría en Ciencias en Ingeniería de Sistemas.

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

Título del estudio:

Un enfoque biobjetivo al problema del reparador.

Número de páginas: 114.

Objetivos y método de estudio: El objetivo principal de este trabajo consiste en el planteamiento y estudio de un problema biobjetivo con diversas aplicaciones potenciales. En este problema se tiene un conjunto de clientes que demanda algún producto o servicio. Se conoce el tiempo de servicio en cada cliente así como los tiempos de viaje entre cada par de clientes se busca encontrar la ruta que visite a todos los clientes partiendo de un depósito y volviendo a él, de forma tal que se minimice tanto la distancia recorrida por el vehículo en la ruta así como el tiempo de espera de los clientes.

Para esto, se propone un modelo matemático biobjetivo en donde cada objetivo

surge a partir de distintos puntos de vista. El punto de vista económico es el objetivo de distancia, en donde se desea minimizar la distancia total recorrida; desde el punto de vista social o de servicio al cliente, se tiene el objetivo de latencia en donde se desea minimizar el tiempo que el cliente espera para recibir el servicio.

Se presenta una metodología de solución al problema, el cual es un algoritmo genético elitista que es muy utilizado en los problemas multiobjetivo gracias a los buenos resultados que ofrece.

CONTRIBUCIONES Y CONCLUSIONES: La contribución de este trabajo se centra en el estudio realizado sobre un problema biobjetivo novedoso, dado que la revisión de la literatura mostró que no existen trabajos donde los objetivos que se buscó optimizar hayan sido tratados conjuntamente.

Es por esto, que tanto el modelo biobjetivo presentado así como el algoritmo propuesto se consideran contribuciones directas de este trabajo.

Firma del asesor: _____

Dra. Ada Margarita Álvarez Socarrás

Capítulo 1

# Introducción

## 1.1 Descripción del problema

En este trabajo se estudia un problema de optimización combinatoria que surge en el contexto de las actividades logísticas de diversas empresas de distribución y servicio. En este problema se tiene un conjunto de clientes que requieren un cierto servicio o producto que debe ser suministrado por la empresa en cuestión. Para suministrar este servicio la empresa cuenta con un agente/vehículo con capacidad suficiente para atender a todos los clientes con una ruta que salga del depósito central, atienda a los clientes y regrese al depósito antes de finalizar la jornada laboral. Desde el punto de vista económico la empresa desea reducir los costos de operación por lo que busca que la distancia total recorrida por el vehículo sea la menor posible.

Por otra parte, la empresa desea mejorar el nivel de servicio a sus clientes, para lo cual busca conseguir que el tiempo que cada cliente espera para recibir el servicio sea el menor posible. Esto es, se requiere diseñar una ruta que visite todos los clientes una sola vez considerando simultáneamente dos criterios de desempeño: minimizar la distancia total recorrida por el vehículo y minimizar la suma de los tiempos de espera de los clientes.

En ambos problemas se tiene un conjunto de clientes que deben ser visitados solamente una vez por un agente/vehículo que sale de un depósito central y debe

regresar a él al culminar la ruta. La diferencia fundamental entre ambos radica en el criterio de desempeño que se considera en cada caso. En el problema del reparador el objetivo es minimizar la suma de los tiempos de espera de todos los clientes (también conocido como minimizar la latencia total), mientras que en el problema del agente viajero el objetivo es minimizar la distancia recorrida por el vehículo.

Debido a que los problemas mono-objetivos involucrados en el problema biobjetivo que se pretende estudiar son clasificados como problemas combinatorios NP-difíciles, este trabajo se dedica al desarrollo de un algoritmo metaheurístico que pueda ofrecer soluciones de calidad en un tiempo de cómputo aceptable.

## 1.2 Hipótesis

El problema del agente viajero y el problema del reparador (también conocido como problema de latencia mínima) son clasificados como problemas NP-completos. El problema bajo estudio es un problema biobjetivo que involucra ambas funciones objetivo, por lo que se vuelve difícil obtener de forma exacta el conjunto de soluciones no dominadas.

Este trabajo se centra en proponer un algoritmo que obtenga una aproximación de este conjunto de soluciones no dominadas en un tiempo computacional aceptable. La hipótesis principal es que el algoritmo que se propone obtiene buenas estimaciones de la frontera eficiente en tiempos de cómputo razonables.

## 1.3 Justificación del estudio

Por separado, tanto el problema del agente viajero como el problema de latencia mínima han sido ya estudiados, sin embargo en la literatura científica hay un número considerablemente menor de trabajos que abordan el problema del reparador.

Hasta el momento, no se conoce de un enfoque biobjetivo en donde la latencia y la distancia de la ruta sean consideradas, si bien la latencia con otro objetivo ó la distancia con otro objetivo sí han sido investigados.

La relevancia de este trabajo está dada principalmente a la investigación de un problema biobjetivo que no ha sido tratado en la literatura científica hasta la fecha. El problema propuesto resulta interesante e importante desde el punto de vista de la optimización y de las aplicaciones prácticas en las que podría ser aplicado, entre las que destacan las situaciones de desastre en donde se busca que los afectados esperen el menor tiempo posible para recibir ayuda, así como la aplicación en alguna empresa en donde no solo se beneficiaría a la empresa encargada de distribuir sus productos sino que se buscaría un beneficio para el cliente, minimizando el tiempo de espera para recibir el servicio o el producto que la empresa está encargada de suministrarle.

## 1.4 Objetivos de la tesis

El objetivo de este trabajo consiste principalmente en el estudio de un problema biobjetivo no estudiado en la literatura científica y proponer un método de solución basado en un algoritmo genético elitista. Para ello se plantearon los siguientes objetivos particulares:

- Revisión bibliográfica donde se aborden problemas biobjetivo que involucren alguno de los criterios considerados en nuestro problema así como el estudio de las metodologías aplicadas en estos problemas biobjetivos.
- Formulación matemática del problema bajo estudio.
- Desarrollo e implementación computacional de una metodología de solución para el problema bajo estudio.
- Análisis de los resultados obtenidos en la experimentación computacional.

## 1.5 Estructura de la tesis

La estructura de este trabajo es la siguiente: en el capítulo dos se describen las características principales del problema del agente viajero así como las características del problema del reparador, que son los dos problemas combinatorios que se encuentran en el centro del problema estudiado en esta tesis. Se hace una revisión del estado del arte y se comentan algunos métodos de solución para cada uno de los problemas asi como algunas de sus aplicaciones. Se describen algunos enfoques biobjetivo que incluyan alguna de las funciones objetivo de los problemas anteriores, así como sus técnicas de solución y una introducción a la optimización multiobjetivo.

En el capítulo tres se da una descripción del problema biobjetivo que se estudia en esta tesis, se presenta el modelo matemático propuesto y el método exacto de resolución que se propone utilizar así como el alcance que tiene el modelo.

En el capítulo cuatro se presenta a detalle el algoritmo del NSGA-II que proponemos para dar solución al problema.

En el capítulo cinco, se presentan algunos resultados obtenidos en la experimentación computacional realizada para evaluar el desempeño del algoritmo propuesto y la comparación que se hizo entre los resultados brindados por el algoritmo exacto y el método que proponemos.

En el capítulo seis, se presentan las conclusiones y el trabajo a futuro.

Capítulo 2

# Marco Teorico

En este trabajo se aborda un enfoque biobjetivo al problema del reparador en el que además de la latencia, el cual es el objetivo usual del problema, se considera también la distancia recorrida por el vehículo en el que se visita a los clientes.

El problema de latencia mínima (o problema del reparador) se considera un problema "centrado en el cliente", debido a que su objetivo consiste en lograr que el cliente espere el menor tiempo posible para recibir un servicio. Esta cualidad lo hace diferente a los problemas centrados en la empresa, como el problema del agente viajero, donde el objetivo está enfocado en minimizar la longitud de la ruta sin considerar en el momento de la toma decisiones, a los clientes.

Con este enfoque biobjetivo se busca no solo beneficiar a la empresa encargada de distribuir sus productos, sino que se busca también, garantizar cierto nivel de servicio al minimizar el tiempo de espera de los clientes para poder recibir el producto o el servicio que la empresa está encargada de suministrarle.

En este capítulo se presentan las definiciones y el material necesario para abordar el estudio del problema que se propone. Hasta el momento, no se conoce de un enfoque biobjetivo similar, en donde la latencia y la distancia de la ruta sean consideradas.

En el problema que se estudia, se debe diseñar una ruta para atender un

conjunto de clientes, partiendo de un depósito y regresando a él al final de la ruta. En el camino, se buscará minimizar tanto el tiempo de espera de cada cliente, como la distancia que recorrerá el vehículo. En este sentido, podemos decir que en este enfoque biobjetivo se conjugan dos problemas importantes: el problema del agente viajero y el problema de latencia mínima.

Es por esto, que en la primera sección se presentarán los conceptos básicos del problema del agente viajero y en la segunda sección los conceptos básicos del problema de latencia mínima. En la tercera sección se presentan algunos enfoques biobjetivo que involucran alguno de los objetivos que nos atañen y finalmente algunos conceptos fundamentales de optimización multiobjetivo así como dos métodos que serán aplicados en nuestro trabajo.

## 2.1 El problema del agente viajero.

El problema del agente viajero (TSP, por sus siglas en inglés) consiste en lo siguiente: teniendo un conjunto de ciudades o clientes así como el costo que representa el viajar de una ciudad a otra, se desea encontrar la ruta de costo mínimo para visitar todas las ciudades, pasando sólo una vez por cada una de ellas y regresando a la ciudad de la cual se partió inicialmente. Este es uno de los problemas de optimización combinatoria más ampliamente estudiado, y aunque existen muchos trabajos y estudios a su alrededor, aún no existe una forma eficiente para resolverlo a optimalidad [20, 48].

Sin embargo, su popularidad se debe a su planteamiento engañosamente simple, su dificultad para ser resuelto [28], así como por sus múltiples aplicaciones en la vida práctica [3]. Algunos ejemplos de estas aplicaciones son los siguientes:

- Perforación de las placas de los circuitos impresos. Para conectar un conductor en una capa con otro conductor en otra capa de la placa o para posicionar

los clips de los circuitos integrados, se deben perforar agujeros a través de dicha placa. Estos agujeros deben ser de diferentes tamaños y para hacer dos agujeros de diámetro distinto, se debe mover el cabezal del taladro hacia la caja de herramientas en donde se cambia la broca del cabezal y regresar a la placa para agujerarla. Esto toma mucho tiempo, y es claro que es mejor hacer todos los agueros del mismo diámetro a la vez, cambiar la broca, hacer los agujeros del diámetro siguiente y así consecutivamente.

Este problema se puede ver como una serie de TSPs, porque para cada diámetro de los agujeros, se puede considerar como las "ciudades" a las posiciones en donde los agujeros tienen que ser realizados y a la "distancia" como el tiempo que le toma al taladro mover el cabezal de una posición a otra. El objetivo es minimizar el tiempo total de viaje del cabezal del taladro [22].

- Turismo y agencias de viajes. Aún cuando los agentes de viajes no tienen un conocimiento explícito del problema del agente viajero, en las empresas donde se encargan de planear viajes, utilizan un software que hace todo el trabajo. Estos paquetes son capaces de resolver instancias pequeñas del TSP, aunque realmente este tipo de clientes utilizan la frase *"La distancia más corta entre dos puntos no es divertida"*[3], así que no buscan tours óptimos realmente.

- Cableado en una computadora. Lenstra y Rinnooy [30], en el año de 1974 reportaron un caso especial en las conexiones de los componentes de una placa de circuitos en una computadora. Los módulos se localizan en dicha placa y un conjunto de clips tiene que ser conectado en ella, sin embargo se debe satisfacer que no más de dos clables se deben unir a cada clip. Se tiene así, un problema de encontrar un ciclo hamiltoniano sin un punto específico de salida y llegada, así como distancias asimétricas.

- Reparación de motores de turbina de gas. Esta aplicación fue reportada en 1987 [38], y se produce cuando los motores de turbina de gas de los aviones tienen que ser revisados.

Para poder garantizar un flujo uniforme del gas a través de las turbinas existen unas guías ubicadas en cada turbina las cuales tienen características individuales y si son colocadas correctamente esto se puede traducir en importantes beneficios, como la reducción de la vibración en la turbina, la uniformidad del fluido a través de ellas, y por consiguiente en un menor consumo de combustible.

El acomodo de las guías puede ser modelado como un TSP con una función objetivo especial.

- Creación de rutas escolares. Esta aplicación fue una de las primeras aplicaciones del TSP. En [3] se menciona que Merrill Flood se interesó en el problema del agente viajero mientras trataba de determinar una ruta escolar óptima. En la actualidad, muchas empresas encargadas del transporte de personas adquieren software de resolución del problema del agente viajero para así reducir significativamente los gastos.

- El reparto de cartas o paquetes, esquema aplicable al caso en que las casas donde se recibirán los paquetes están muy distanciadas entre ellas o cuando solo se deben visitar algunas de ellas.

- El reparto de bienes o servicios, entre otros.

Estas aplicaciones del problema del agente viajero no son las únicas, existen muchísimas más, es por esto que el problema del agente viajero es uno de los problemas de optimización combinatoria más estudiados actualmente.

El problema del agente viajero se define sobre un grafo completo $G = (V, A)$, donde $A$ es el conjunto de los arcos y $V=\{1, 2, 3, \ldots, n\}$, es el conjunto de nodos. Se tiene una matriz $C = c_{ij}$ que puede ser interpretada como la matriz de costos o la matriz de tiempos de viaje de la ciudad $i$ a la ciudad $j$. Si consideramos las variables de decisión $x_{ij}$ como variables binarias donde

$$x_{ij} = \begin{cases} 1, & \text{Si el arco que va de la ciudad } i \text{ a la ciudad } j \text{ es utilizado en la solución,} \\ 0, & \text{en caso contrario.} \end{cases}$$

La formulación matemática del problema del agente viajero, es la siguiente:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{s.a} \quad \sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \ \forall j \tag{2.1}$$

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} x_{ij} = 1, \ \forall i \tag{2.2}$$

$$\sum_{\substack{(i,j) \in A \\ i \in S \\ j \in V \setminus S}} x_{ij} \geq 1, \ \forall S \subset V \tag{2.3}$$

$$x_{ij} \in \{0, 1\} \tag{2.4}$$

En esta formulación, (2.1) y (2.2) son las restricciones que garantizan que solo se entra y se sale una vez de cada ciudad, cada una de estas restricciones tiene una cardinalidad de $n$. La restricción (2.3), es la restricción que garantiza que no habrá sub-tours en la solución, es decir, es la restricción que garantiza que no haya recorridos que no pasen por el depósito. Realmente, esta es la restricción que complica el modelo al tener un crecimiento exponencial, ya que se tiene una restricción por cada subconjunto del conjunto de vértices. Esta restricción tiene una cardinalidad $2^n$.

Debido a la esencia del problema, muchas aplicaciones prácticas son evidentes y en todas ellas, el costo o la distancia entre cada par de ubicaciones, ya sean ciudades, nodos en una red, etcétera, es conocido. En general, el número de soluciones de un problema de agente viajero es del orden de $n!$. En la figura 2.1, se pueden observar

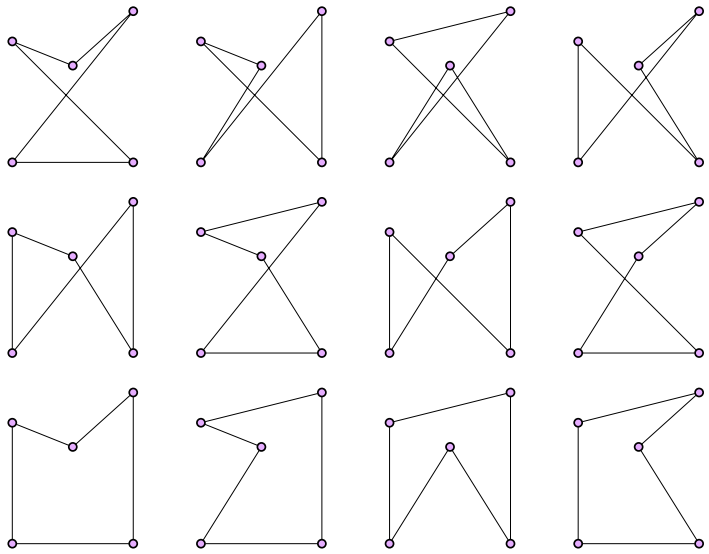


Figura 2.1: Ejemplo de diversas soluciones de un TSP.

diferentes soluciones de un TSP de 5 nodos, sin embargo éstas no son todas las soluciones posibles a este problema en específico.

Algunos trabajos que reflejan el amplio estudio que ha tenido este problema son, Laporte [29], Goyal [21], Johnson y McGeoch [16], Cirasella et al. [13], Orman et al. [36], entre otros. Otras aplicaciones reales, se pueden encontrar en Dorigo y Gambardella [18], Carpaneto y Toth [10], entre otros.

El problema del agente viajero puede ser resuelto de varias formas, una de ellas es llamada enumeración de todas las soluciones o *fuerza bruta*. Este método siempre encuentra la solución óptima pero sacrificando muchísimo tiempo al ir explorando todas las soluciones al problema. Cuando se tiene una instancia grande, las soluciones que deben ser estudiadas son muchísimas ya que el problema tiene un crecimiento exponencial y por lo tanto el tiempo que tarda en ser resuelto es demasiado.

Existen los *métodos exactos*, como el método de ramificación y acotamiento, los cuales aseguran matemáticamente que se encuentra la solución óptima pero con la misma desventaja, sacrificando demasiado tiempo bajo las mismas condiciones.

Es por esto, que en la práctica, se utilizan *heurísticas*. A continuación se describen las heurísticas más comunes para resolver el problema del agente viajero.

**Heurística del Vecino más Cercano.**

La heurística del vecino más cercano es la heurística más sencilla y la más natural de todas, apartir de un vértice inicial seleccionado aleatoriamente, el siguiente vértice en el tour será el vértice más cercano a este vértice inicial. Así se continúa el proceso, seleccionando para añadir al tour en construcción el vértice más cercano al último añadido hasta que todos los vértices han sido agregados. Se dice que esta es la heurística más natural de todas porque se basa en la idea de moverse a la ciudad siguiente de la forma que le resulte más barata al agente viajero y respeta los lineamientos del problema al no poder visitar ciudades por más de una ocasión y al siempre regresar a la ciudad de donde se parte.

La importancia y popularidad de este método se debe a su simplicidad, sin embargo, se ha observado que en general tiene un pobre desempeño.

**Heurística de inserción.**

Otra heurística para resolver el problema del agente viajero consiste en comenzar con un tour incompleto e insertar los nodos faltantes de tal forma que el costo del tour sea el mejor posible.

Existen varios criterios para insertar los vértices (Figura 2.2) en el algoritmo de inserción, esto obviamente ligado al criterio de selección del vértice que será insertado en el tour parcial.

Se define además, la distancia de un vértice $v$ al ciclo como el mínimo de las distancias de $v$ a todos los vértices del ciclo o tour parcial, esto es:

$$d_{min}(v) = min\{C_{iv}/i \in V \backslash W\}$$

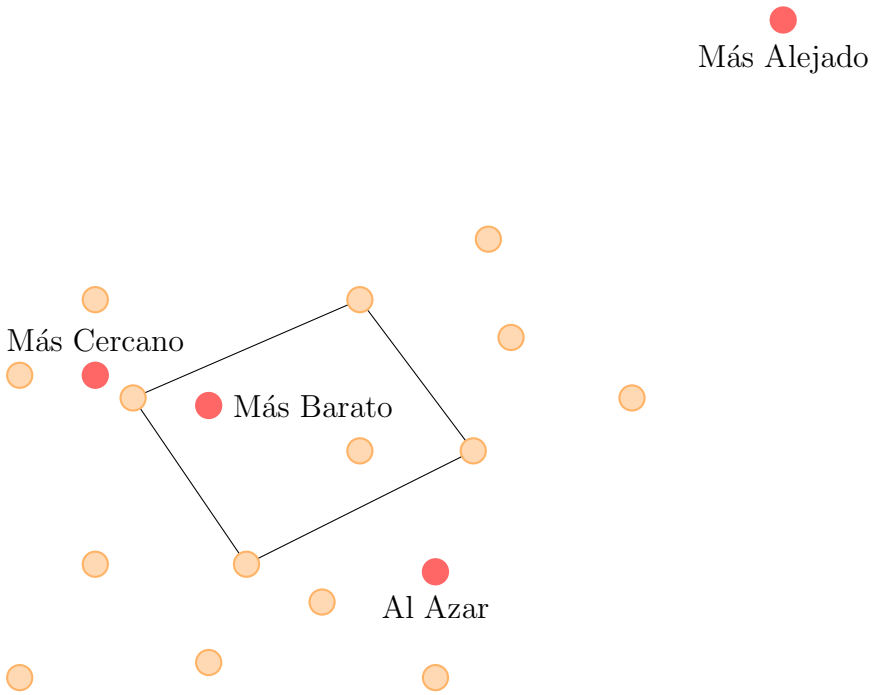



Figura 2.2: Figura que muestra diferentes criterios de inserción.

Los criterios de inserción, pueden ser:

- Inserción más cercana: Consiste en seleccionar el vértice $j$ más cercano al tour incompleto que se tiene. Los vértices seleccionados se insertan al final.

$$d_{min}(j) = min\{d_{min}(v)/v \in W\}$$

- Inserción más lejana. La cual consiste en seleccionar el vértice $j$ más alejado del ciclo inicial e insertarlo al final.

$$d_{min}(j) = max\{d_{min}(v)/v \in W\}$$

- Inserción más barata. Consiste en seleccionar el vértice $j$ que genere el menor incremento de costo insertándolo al final del ciclo inicial. La diferencia entre

esta inserción y la inserción más cercana consiste en que mientras en la inserción más barata se considera el costo del tour completo, en la inserción más cercana se considera nada más el costo de ciudad a ciudad.

- Inserción aleatoria. El vértice a insertar $j$, es seleccionado al azar e insertado al final del ciclo inicial.

Es importante mencionar que el problema del agente viajero se puede considerar como un subproblema de otros problemas de optimización combinatoria [21], tales como el problema de ruteo de vehículos.

## 2.2 El problema de mínima latencia.

Un problema similar al del agente viajero, aunque no realmente derivado directamente de él [48], es el problema de latencia mínima (MLP, por sus siglas en inglés). En este problema un proveedor de servicios (como un repartidor o un reparador de bienes), debe atender un conjunto de peticiones cada una de las cuales se considera un cliente. Este proveedor desea arreglar el orden en que visita a sus clientes de tal forma que se minimice el tiempo total que los clientes deben esperarlo antes de ser atendidos [8].

A este problema se le conoce también como el *problema del repartidor* o como el *problema del reparador* [47, 32], en ambos problemas se conocen de antemano la ubicación de los clientes así como los tiempos de servicio y el objetivo en ambos casos es encontrar la ruta que minimice el tiempo total de espera de los clientes que el agente debe visitar. La diferencia entre el problema del reparador y el problema del repartidor consiste en que los tiempos de entrega que se manejan en el problema del repartidor son menores que los tiempos de servicio de reparación del problema del reparador [1].

El problema de latencia mínima se define sobre un grafo completo $G = (V, A)$, donde $A$ es el conjunto de los arcos y $V = \{0, 1, \ldots, n\}$ es el conjunto de vértices. El nodo 0 es el nodo raíz o depósito, y la matriz $C_{ij}$ es una matriz de pesos no negativos asociados a los arcos. Si denotamos el tiempo de servicio al cliente $i$ como $s_i$ y al tiempo de viaje del cliente $i$ al cliente $j$ como $t_{ij}$, podemos entonces definir la matriz de costos como

$$C_{ij} = s_i + t_{ij}$$

Dada una permutación $P = \{0, [1], [2], \ldots, [n]\}$, la latencia del nodo $i$ se define como la longitud del camino desde el nodo raíz hasta $i$ como

$$l_i = \sum_{j=1}^{i} C_{[j-1][j]} \tag{2.5}$$

En la expresión (2.5), se denota por $[i]$ al índice del nodo en la posición $i$ en el camino. Se tiene además que el índice del nodo raíz es [0] y que la latencia del nodo raíz es cero: [0]=0, $l_{[0]} = l_0 = 0$.

Para ejemplificar, suponga que un reparador de aparatos electrónicos está encargado de visitar a cuatro clientes, saliendo de su establecimiento y desplazándose hacia ellos. En la figura 2.3, se puede observar un posible orden de los clientes para ser atendidos; el costo está definido como $C_{ij} = s_i + t_{ij}$, es decir, está representado como la suma del tiempo de servicio al nodo $i$ y el tiempo de desplazamiento del nodo $i$ al nodo $j$.

En la figura 2.4, aparece el costo que representa viajar del depósito al primer cliente. Este costo, está dado por el tiempo que se tarda el reparador en cubrir el servicio del nodo depósito más el tiempo de traslado desde el nodo depósito hasta el cliente uno. Como el depósito es el nodo raíz, su latencia es cero, de forma que el costo $C_{D,1} = t_{D,1}$.

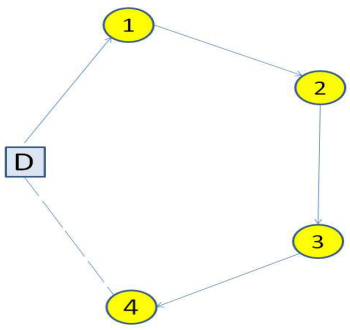



Figura 2.3: $C_{ij} = s_i + t_{ij}$

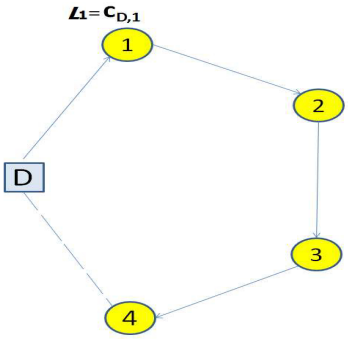



Figura 2.4: $C_{D,1} = s_D + t_{D1}$, $s_D = 0 \Rightarrow C_{D,1} = t_{D,1}$

En la figura 2.5 se representa el costo que se tiene de viajar del cliente uno al cliente dos. Sin embargo, la latencia real en el nodo dos está dada por el costo que se tuvo al viajar y atender al cliente uno más el costo que se tiene al viajar al cliente dos. Es decir, el tiempo total que tiene que esperar el cliente dos en ser atendido está dado por:

- El tiempo de servicio en el depósito más el tiempo de traslado del depósito al cliente uno sumado a,
- El tiempo de servicio al cliente uno más el tiempo de traslado del cliente uno al cliente dos.

$$L_2 = t_{D1} + (s_1 + t_{12}) = C_{D,1} + C_{1,2}$$

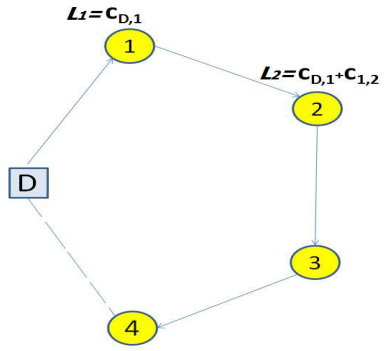



Figura 2.5: $l_2 = C_{D,1} + C_{1,2}$

En este ejemplo de 4 clientes, la latencia total de la ruta está dada por:

$$\begin{aligned} C_{D,4} &= L_1 + L_2 + L_3 + L_4 \\ &= (C_{D,1}) + (C_{D,1} + C_{1,2}) + (C_{D,1} + C_{1,2} + C_{2,3}) + (C_{D,1} + C_{1,2} + C_{2,3} + C_{3,4}) \\ &= 4 * C_{D,1} + 3 * C_{1,2} + 2 * C_{2,3} + 1 * C_{3,4} \end{aligned}$$

Se puede observar, que el tiempo total que tiene que esperar el cliente cuatro incluye todos los tiempos de viaje de los clientes anteriores a él. Sin embargo, el tiempo de regreso al depósito no se considera en ningún lugar, esto se debe a que el regreso al depósito no afecta los tiempos de espera de los clientes, por lo cual no es considerado.

Tomando este ejemplo, se puede decir que la latencia total para una permutación $P$ es la suma de la latencia de todos los nodos y es calculada de la siguiente manera,

$$\begin{aligned} L &= \sum_{i=1}^{n} l[i] \\ &= \sum_{i=1}^{n} (n - i + 1) C_{[i-1][i]} \end{aligned} \tag{2.6}$$

En resumen, el problema de latencia mínima consiste en encontrar la permutación con el menor valor de la latencia total entre todas las permutaciones

de nodos de $V$ que tengan al nodo raíz en la primera posición. De la expresión (2.6), se puede observar que la contribución de cada cliente a la función objetivo depende de la posición que este tenga en la ruta. Nótese que en el problema del reparador, a diferencia del problema del agente viajero, no se considera el arco de regreso. Esto tiene sentido, porque a el reparador le interesa minimizar el tiempo que sus clientes deben esperarlo antes de ser atendidos y no el tiempo que él tarda en regresar a su establecimiento.

Existen distintas formulaciones reportadas para el problema de latencia mínima entre las que destancan las formulaciones de Méndez-Díaz et al [33], Gouveia y Voss [20], asi como el modelo propuesto por Picard y Queyranne [37] e implementado por Sarubbi [43]. Sin embargo, en el 2012 Angel-Bello et al [1] propusieron dos modelos que tienen un mejor desempeño que los modelos anteriores.

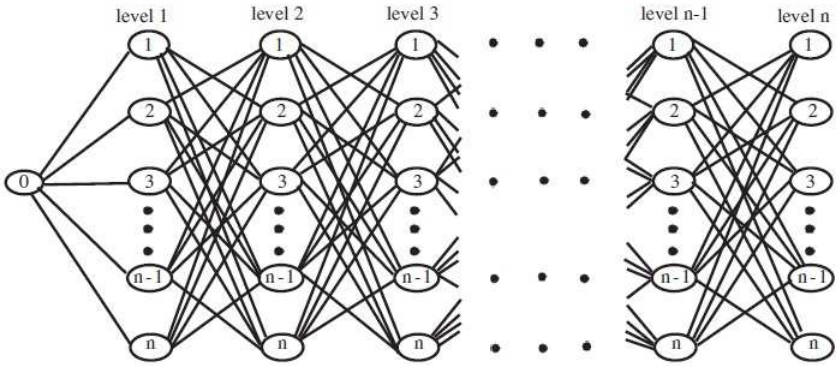



Figura 2.6: Red Multinivel propuesta por Picard y Queyranne.

Ambos modelos fueron formulados sobre una red multinivel de $n + 1$ niveles, la cual está basada en una red anteriormente propuesta por Picard y Queyranne [37] y puede observarse en la figura 2.6. En esta red, el nivel 0 se refiere al depósito central, o el lugar de donde el agente debe partir y volver una vez que ha visitado a todos los clientes. Los niveles $1, 2, \ldots, n$, son $n$ copias de los $n$ clientes que el agente debe visitar. Dado a que la solución buscada es una permutación, ésta puede ser

representada en dicha red, escogiendo solamente un cliente en cada nivel el cual no puede estar elegido en ningún otro nivel.

Las aplicaciones del problema de mínima latencia son muy variadas, entre las que se encuentran las siguientes:

- Logística humanitaria. Es una rama de la logística, la cual se especializa en la entrega y almacenamiento de recursos destinados para algun área afectada durante un desastre natural, una catástrofe o una emergencia. A pesar de que la logística ha sido mayormente utilizada en las cadenas de suministro comerciales, es una de las herramientas más importantes actualmente en las operaciones de auxilio en una situación de desastre.

  Específicamente, las actividades de planeación, implementación y control, así como el mantener un costo eficiente en el flujo y almacenamiento de los recursos y materiales, así como también de la información, desde un punto origen hasta un punto receptor, esto con el propósito de aliviar el sufrimiento de gente vulnerable, se conoce como "logística humanitaria"[47].

  En un desastre, es importante asegurar el envío eficiente y efectivo de los recursos, de forma que estos lleguen a las personas afectadas por la emergencia de la forma más rápida posible [14]. En un desastre se busca que las personas afectadas esperen el menor tiempo posible por la ayuda, premisa principal en el problema del repartidor.

- La recepción y recuperación de datos en redes de computadoras e internet donde se busca minimizar el tiempo de espera del usuario a páginas web [31].

- La calendarización de trabajos, problema donde se tiene una lista de trabajos que deben ser procesados en una sola máquina, la cual puede verse como el depósito y los trabajos como los clientes. El objetivo es minimizar la suma de los tiempos que los trabajos tardaron en finalizarse [12].

- La modificación de la información a través de redes inalámbricas multi-salto. Este problema se define de la siguiente manera: a cada nodo en la red se le asigna inicialmente un mensaje y el objetivo es diseñar un calendario de tal forma que cada nodo distribuya su mensaje a todos los otros nodos pero de tal forma que cada nodo espere el menor tiempo posible para recibirlo [24].

- Reparación o reparto de artículos. Es la aplicación natural del problema del reparador, se tiene un repartidor o un reparador y un conjunto de clientes a los cuales se debe atender de forma tal que cada cliente espere el menor tiempo posible.

Algunos artículos que reflejan el estado del arte son Blum et al [8], Wu et al [48], Angel-Bello [1], Silva et al [45], Archer y Blasiak [4], Li et al [32], Huang et al [25], Qadir et al [39], Salehipour et al [42], Friggstad et al [19], entre otros.

El problema del reparador puede ser resuelto de diferentes maneras. Al igual que el problema del agente viajero, se puede encontrar la solución enumerando todas las soluciones posibles y seleccionando la mejor de todas, sin embargo este método tarda tanto tiempo que no es una vía práctica. Existen también los métodos exactos, como el método de ramificación y acotamiento, los cuáles aseguran matemáticamente que se encuentra la solución óptima pero con la misma desventaja, sacrificando demasiado tiempo bajo las mismas condiciones.

Es por esto, que en la práctica, se utilizan métodos heurísticos o metaheurísticos así como relajaciones para mejorar algún método exacto existente. A continuación se describen algunos métodos que han sido utilizados para resolver el problema del reparador.

En el trabajo de Chakrabarty publicado en el 2011 [11], se introduce un problema considerado como una generalización de los problemas de ubicación de instalaciones no capacitado (UFL) y el problema de latencia mínima (MLP) en

donde las instalaciones no solo necesitan ser abiertas para satisfacer las demandas de los clientes, sino que también deben ser activadas secuencialmente antes de que puedan proveer el servicio. Este trabajo se centra en un enfoque Lagrangiano como un método para acelerar la resolución de la relajación lineal por medio del método simplex, se prueban desigualdades y diferentes cotas para así enfatizar que la técnica de relajaciones lineales es flexible y no tarda demasiado.

En el 2003, Archer et al [5] desarrollan un algoritmo aproximado para este problema. Un algoritmo $\alpha$-aproximado produce soluciones con una latencia total no mayor a $\alpha$ veces la latencia total del tour con la mínima latencia. A este valor $\alpha$ se le llama "garantía de ejecución del algoritmo". El primer algoritmo $\alpha$-aproximado fue presentado por Blum et al [8] y alcanzó una garantía de ejecución de 72. En el trabajo de Archer et al, se explora la conexión entre el problema de latencia mínima y el problema del mínimo k-árbol de expansión, el cual busca un arbol de costo mínimo que tenga exactamente $k$ vértices. En general, se obtiene un algoritmo con una garantía de 7.18, la cual mejora un poco el trabajo de Blum et al, tanto en la garantía como en el tiempo de la ejecución.

En el 2011 se presenta la primer metaheurística para resolver el problema de latencia mínima. Salehipour et al [42] presentan un GRASP+VND y un GRASP+VNS los cuales son métodos multi-inicio en los que cada iteración consiste de dos fases: una fase de construcción greedy aleatoria (GRASP) y una fase de mejora (VND o VNS).

La característica más importante del GRASP es la forma en la que combina aleatoriedad y voracidad en la fase constructiva. Para este fin, se construye una lista restringida de candidatos, asumiendo que el problema es de minimización, de forma que contenga los elementos que lleven al menor incremento en el valor de la función objetivo. De la lista restringida se selecciona un elemento de forma aleatoria y se actualiza para reflejar el hecho de que un nuevo elemento fue agregado a la solución

y no está disponible para ser seleccionado nuevamente. Este proceso de selección y actualización de la LRC se repite hasta que una solución completa ha sido creada. Es de esta solución, donde se parte para la siguiente fase que es una búsqueda local y así encontrar un óptimo local.

En la segunda fase, se utilizan dos metaheurísticas para explorar sistemáticamente diferentes estructuras de vecindarios, el VND o Variable Neighborhood Descent y el VNS o Variable Neighborhood Search, los cuales tienen como idea principal, que un óptimo local en una vecindad no necesariamente es el mismo óptimo local en una vecindad diferente, y por esta razón se puede escapar de un óptimo local cambiando la estructura del vecindario. La diferencia entre un algoritmo y otro, es que el VNS utiliza un movimiento llamado "permutación" que tiene como objetivo perturbar la solución actual de forma que su estructura sea parcialmente destruída.

Los algoritmos VNS y VND examinan todas las vecindades comenzando en las pequeñas y terminando con las grandes, esto es, porque las vecindades pequeñas contienen menos soluciones y pueden ser investigadas en menor tiempo que las vecindades grandes, las cuáles sólo son examinadas cuando todas las vecindades pequeñas han sido agotadas, es decir, cuando la solución actual es un óptimo local relativo a todas las vecindades pequeñas. El algoritmo VND se detiene cuando el óptimo local de todos los vecindarios es la solución actual.

En general, el problema de mínima latencia ha sido resuelto ya sea con métodos exactos los cuales en gran escala son insuficientes, de forma aproximada con algoritmos de aproximación y con algoritmos heurísticos, teniendo un mejor desempeño estos últimos.

## 2.3 Algunos enfoques biobjetivo

Por separado, tanto el problema del agente viajero como el problema de latencia mínima han sido ya estudiados y se encuentran numerosos trabajos en la literatura científica aunque en menor cantidad para el problema de latencia mínima. En un contexto biobjetivo, latencia y distancia no han sido estudiados, sin embargo, latencia con otro objetivo, ó distancia con otro objetivo, sí han sido investigados. Algunos trabajos de estos enfoques biobjetivos reportados en la literatura son los siguientes.

Un ejemplo del uso de la función objetivo de distancia en un problema biobjetivo, es el trabajo publicado en el 2009 por Bérubé et al [9], quienes aplicaron el método de la $\epsilon$-restricción para encontrar el frente exacto del problema del agente viajero con ganancias, el cual es una variante del problema del agente viajero en donde un beneficio o una ganancia se asocia a cada vértice. El objetivo del trabajo es maximizar los beneficios obtenidos y minimizar el costo del viaje.

Otro ejemplo es el trabajo de Riera-Ledesma [40], publicado en el 2005. En este trabajo se busca resolver el problema biobjetivo del comprador viajero, el cual consiste en determinar una ruta a través de un conjunto de mercados para conseguir un conjunto de productos buscando minimizar la distancia recorrida y el costo de las compras de forma simultánea. Este problema está formulado como un modelo biobjetivo de programación lineal entera mixta con un número exponencial de desigualdades válidas y es resuelto con un algoritmo de planos cortantes para generar el conjunto de todos los puntos eficientes soportados, y los no soportados, en el espacio objetivo. Para cada punto eficiente encontrado en el espacio de los objetivos, una solución Pareto óptima es calculada resolviendo un problema mono objetivo con el algoritmo de ramificación y corte.

En el 2008, Jozefowiez et al [26], desarrollaron un trabajo donde se busca

minimizar la longitud de la ruta y la maximización de las ganancias. Ellos emplean una búsqueda local combinada con un algoritmo evolutivo multi-objetivo, el cual lo utilizan para generar diversidad en las soluciones iniciales obtenidas y obtener mejores soluciones al final del proceso.

Los estudios que se han hecho con el objetivo de minimizar tiempos de espera son aún más escasos, y principalmente se centran en la secuenciación de tareas en máquinas. Tal es el caso del artículo de Tavakkoli-Moghaddam, Rahimi-Vahed y Hossein Mirzaei [46] del año 2007. En este artículo utilizan un algoritmo híbrido basado en un sistema inmune y la optimización bacterial para encontrar soluciones óptimas de Pareto tomando como objetivos minimizar el tiempo medio ponderado de terminación de las tareas y la tardanza media ponderada.

Sin embargo, en el 2010 Chakrabarty y Swamy [11] trabajaron una variante del problema de mínima latencia, ahora enfocados a la localización de instalaciones, donde éstas no sólo necesitan ser abiertas para atender a los clientes, sino que además necesitan ser activadas secuencialmente antes de que puedan brindar servicio a los clientes. Este trabajo se enfoca en tres poblemas ampliamente estudiados en la investigación de operaciones, la ubicación de las instalaciones, debido a que se busca de un conjunto de ubicaciones el abrir instalaciones que conecten a los clientes a un centro abierto y minimizar el costo de apertura innecesaria. El problema de ruteo de vehículos, dado que se considera el reparto de producto a domicilio de los clientes y el problema de mínima latencia, ya que quieren proveer el servicio al cliente de la forma más rápida posible. Sin embargo, los dos objetivos que se consideran son minimizar la longitud de las $k$-rutas de los vehículos que llevan el producto y, adoptando un enfoque orientado al cliente, el minimizar la suma de las latencias de los clientes. Chakrabarty y Swamy, realizaron su investigación directamente sobre el problema de ubicación de facilidades y el problema de mínima latencia, aún cuando mencionan que la longitud de las rutas y la latencia serán los objetivos, tomando la distancia como un parámetro y resolviendo relajaciones lineales.

En el 2008, Benoit et al [7], consideraron la aplicación del problema de latencia mínima al problema de mapeo en plataformas heterogéneas donde una aplicación típica es el procesamiento de imágenes digitales. Un ejemplo de este tipo de aplicaciones es la codificación de imágenes JPEG. Este tipo de aplicaciones representan un gran desafío debido a que los procesadores tienden a fallar durante su ejecución.

Este trabajo en particular estudia la complejidad del mapeo bi-criterio, el cual tiene como objetivo optimizar la latencia, es decir, el tiempo de respuesta de la aplicación, y maximizar la confiabilidad, es decir, aumentar la probabilidad de que la ejecución de la aplicación sea exitosa.

En esta aplicación, una serie de datos o tareas son procesadas al inicio de la aplicación para ser consideradas a lo largo del proceso hasta que éste ha sido completado. Cada etapa del procesador tiene sus requerimientos computacionales, tiene que leer las entradas introducidas en la etapa anterior, procesar la información y generar una salida para la etapa siguiente. La latencia se minimiza utilizando procesadores más rápidos, mientras que la confiabilidad aumenta haciendo réplicas del proceso. Sin embargo, las réplicas aumentan la latencia. Es por esto que se busca un compromiso entre la latencia y la probabilidad de falla.

En el 2012, Gupta et al [23] buscan minimizar el número de trabajos atrasados mientras que también buscan minimizar el tiempo total de la completitud de las tareas. Estos objetivos son optimizados secuencialmente, primero utilizando el tiempo total de completitud y después minimizando el número de trabajos atrasados, esto sujeto a la optimización anterior.

Sin embargo, artículos donde se estudien los objetivos de minimizar distancia y minimizar latencia simultáneamente, no hemos encontrado.

## 2.4 Optimización multiobjetivo

El enfoque biobjetivo propuesto traería beneficios no sólo a la empresa sino que el cliente tomaría un papel importante a la hora de tomar una decisión. A continuación, se expondrán algunos elementos básicos de optimización multiobjetivo necesarios para resolver el enfoque propuesto.

La mayoría de los problemas de decisión reales implican la optimización simultánea de más de un objetivo, es por esto que la optimización clásica presenta importantes debilidades que la desvía considerablemente de los procesos reales de toma de decisiones. En muchos casos de la vida ordinaria, el tomador de decisiones no está interesado en ordenar las soluciones factibles en base a un único criterio, si no que desea efectuar esta tarea basado en diferentes criterios que reflejen sus preferencias.

En estos casos, generalmente no existe una alternativa que sea la mejor opción para todos los criterios al mismo tiempo y se requiere un análisis para determinar qué alternativa o cuáles alternativas son las más adecuadas para el problema. La programación multiobjetivo, es la rama de la programación matemática que se encarga de dar solución a ese tipo de problemas.

La programación multiobjetivo constituye un enfoque multicriterio de gran potencialidad cuando el contexto decisional está definido para una serie de objetivos a optimizar. Como la optimización simultánea de todos los objetivos es usualmente imposible, debido al grado de conflicto que por lo general existe entre los objetivos, el enfoque multiobjetivo en lugar de intentar determinar un óptimo, pretende establecer un conjunto de soluciones eficientes [41]. La estructura de un problema multiobjetivo es

$$Eff\ \ F(x) = [f_1(x), \ldots, f_i(x), \ldots, f_q(x)]$$

sujeto a:

$$x \in X$$

Donde,

- *Eff*, significa la búsqueda de soluciones eficientes u óptimas en el sentido de Pareto,
- $F_i(x)$, son las expresiones que denotan los objetivos,
- $x$, es el vector de variables de decisión,
- $X$, es el conjunto de restricciones que definen el conjunto de soluciones posibles.

El propósito de un enfoque multiobjetivo es determinar un subconjunto propio del conjunto de todas las soluciones posibles, cuyos elementos sean "óptimos". Esto se logra utilizando estrictamente información técnica, sin tomar en cuenta las preferencias del centro decisor. El conjunto $Y = F(X)$ corresponde al espacio imagen, es decir, a las soluciones factibles en el espacio de objetivos y $y = (y_1, y_2, \ldots, y_p)$, $y_i = f_i(x)$ es una solución en el espacio de objetivos.

Sin embargo, ¿cómo saber cuál solución es más deseable que otra? Esto se complica al ya no tener una función objetivo, sino un vector de funciones objetivo. Es aquí donde se introduce un orden inspirado en el concepto de eficiencia, el cual fue desarrollado por Vilfredo Pareto en 1896. En su formulación inicial, Pareto considera que una solución se encuentra en un estado óptimo si ningun atributo de esa solución puede mejorar sin empeorar a alguno otro. El concepto de optimalidad paretiana puede definirse de la siguiente manera: Un conjunto de soluciones es *eficiente ó Pareto-Óptimo* cuando está formado por soluciones factibles, tales que no existe una solución factible que proporcione una mejora en un atributo sin producir un empeoramiento en algún otro. Matemáticamente, esto es:

Una solución $y = (y_1, y_2, \ldots, y_p)$ domina ($\prec$) a una solución $z = (z_1, z_2, \ldots, z_p)$ si y sólo si, $\forall i \in \{1, 2, \ldots, p\}$, $y_i \leq z_i$ y existe un $j \in \{1, 2, \ldots, p\}$ para el cual $y_j < z_j$. Una solución Pareto óptima es aquella solución que no es dominada. Todos los enfoques multicriterio pretenden obtener soluciones que sean eficientes en el sentido paretiano anteriormente definido.

Un primer paso en la optimización multiobjetivo, consiste en la obtención de la matriz de pagos. Esta matriz permite cuantificar el nivel de conflicto existente entre los objetivos que se están considerando.

La matriz de pagos se obtiene optimizando cada objetivo por separado y calculando los valores alcanzados por los demás objetivos en cada solución óptima. La matriz de pagos resultante es una matriz cuadrada cuya dimensión coincide con el número de objetivos que se busca optimizar. Los elementos de la diagonal principal de esta matriz corresponden a lo que suele denominarse *punto ideal*, es decir, es en esta solución en la que todos los objetivos alcanzan su valor óptimo. En la vida real, los objetivos a considerar entran en conflicto entre sí, por lo cual el punto ideal es prácticamente inalcanzable, sin embargo es de gran utilidad en el desarrollo de diversos métodos multiobjetivo. A la solución conformada considerando el peor elemento de cada columna de la matriz de pagos se le conoce como *punto anti-ideal*, y en general, representa una solución poco atractiva para el centro decisor, sin embargo, es muy útil conocerlo porque con él, se pueden normalizar los objetivos que están medidos en unidades diferentes. Además al conocer tanto al punto ideal como al punto anti-ideal, se define un intervalo de valores para cada atributo. Este intervalo es de gran utilidad en algunas técnicas multiobjetivo, como es el caso del método de la $\epsilon$-restricción.

### 2.4.1 El método de la $\epsilon$-restricción

Existen diversos métodos capaces de brindar el frente exacto de Pareto. Uno de estos métodos es el llamado $\epsilon$-constraint o método de la $\epsilon$-restricción, método que además es uno de los más utilizados en la programación multiobjetivo.

El método de la $\epsilon$-restricción, en el caso biobjetivo, consiste en optimizar sólo un objetivo considerando al otro objetivo como una restricción acotada por un cierto valor $\epsilon$, el cual varía dentro del intervalo obtenido con los puntos ideal y el anti-ideal. Es decir, si se tiene el problema de

$$\begin{aligned} &Minimizar\ f_1 \\ &Minimizar\ f_2 \\ \text{sujeto a} \quad &x \in X \end{aligned}$$

entonces, el modelo a resolver con el $\epsilon$-constraint es el siguiente:

$$\begin{aligned} &Minimizar f_1 \\ \text{sujeto a} \quad &f_2 \leq \epsilon \\ &x \in X \end{aligned}$$

ó

$$\begin{aligned} &Minimizar f_2 \\ \text{sujeto a} \quad &f_1 \leq \epsilon \\ &x \in X \end{aligned}$$

Este modelo, se resuelve para varios valores de $\epsilon$. El valor de $\epsilon$ al que está sujeta la restricción de $f_2$, tiene como límite superior al punto anti-ideal y como límite

inferior al punto ideal, ambos obtenidos con la matriz de pagos. Esto tiene sentido ya que el punto anti-ideal es lo peor que se puede obtener en la función objetivo $f_2$, ya que corresponde a la evaluación en el objetivo $f_2$ del óptimo de $f_1$, el punto ideal es el óptimo de $f_2$, es decir, el mejor valor que se puede obtener en $f_2$. Es por esto, que el épsilon sólo se recorre del punto anti-ideal, es decir, de su peor valor, hasta el punto ideal, que es el mejor valor que puede tomar.

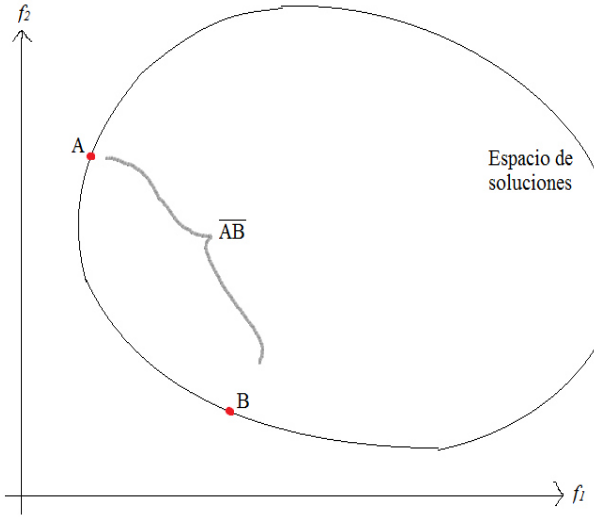



Figura 2.7: Épsilon-constraint

En la figura 2.7, se puede constatar lo anterior. Al resolver a $f_1$ a optimalidad, se acota a $f_2$ en el intervalo dejado entre el punto A (el cual representa al punto anti-ideal de la matriz de pagos), y el punto B (el cual representa al punto ideal de la matriz de pagos). A este segmento se le denominó $\overline{AB}$ en la figura. El método de la $\epsilon$-restricción es un método iterativo, el épsilon se actualiza iteración a iteración para que abarque todo el intervalo desde el punto anti-ideal hasta el ideal, esto se realiza con un salto en el épsilon definido por el usuario. En la imagen 2.8 se puede observar un ejemplo de solución del método.

Debe señalarse, que el método de la $\epsilon$-restricción garantiza la generación de

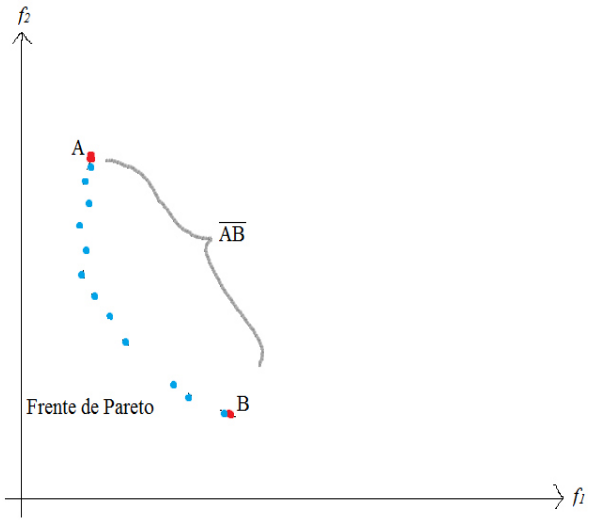



Figura 2.8: Ejemplo de frontera de Pareto obtenida mediante el método del $\epsilon$-constraint

soluciones eficientes sólo cuando las restricciones son activas en el óptimo, es decir, cuando la restricción se satisface como igualdad. Por el contrario, si en el óptimo alguna de las restricciones no es activa y además existen otros óptimos alternativos, la solución generada por el método puede no ser eficiente.

### 2.4.2 NSGA II

Así como puede obtenerse el frente exacto de Pareto por medio de un algoritmo exacto, puede buscarse una aproximación al frente de Pareto por medio de algún método heurístico o metaheurístico, es decir, por medio de algún método que brinde buenas soluciones cercanas al frente exacto pero con un menor tiempo de cómputo.

La metaheurística NSGA-II fue desarrollada por Debb et al en el 2000 [17]. Es un algoritmo genético elitista que es muy utilizado en los problemas multiobjetivo gracias a los buenos resultados que ofrece. Este algoritmo opera sobre un conjunto de soluciones, una población de soluciones, y usa operadores típicos de los algoritmos

genéticos: Selección de padres, cruce, mutación, selección de individuos que pasan a la siguiente generación, etcétera.

Esta metaheurística ha sido utilizada en diferentes problemas de optimización combinatoria. Algunos ejemplos de trabajos encontrados en la literatura utilizando alguna de las funciones objetivo bajo estudio, son los siguientes:

En la función objetivo de distancia, se tiene un estudio biobjetivo del problema del agente viajero con ganancias. En este trabajo del 2008, Jozefowiez et al [27] buscaron minimizar la longitud de la ruta así como maximizar la recolección de beneficios y utilizaron una modificación del algoritmo NSGA-II con la que se obtuvieron buenos resultados.

Bajo la función objetivo de latencia, se tiene el trabajo del 2012 de Susmita Bandyopadhyay [6], en este trabajo se estudia un problema de secuenciación de máquinas en donde los objetivos que se busca optimizar son la minimización del tiempo de finalización de las tareas así como la minimización del costo de penalización por no finalizar la tarea. Para la resolución de este problema se utilizó un algoritmo NSGA-II y se obtuvieron buenas aproximaciones al frente exacto.

Debido a los resultados positivos que se han encontrado en la literatura, en este trabajo se propone una implementación de la metaheurística NSGA-II para el problema bajo estudio. Esta implementación será explicada a detalle en el capítulo tres.

Capítulo 3

# Descripción y modelación del problema

En este capítulo se describe formalmente el problema bajo estudio y se desarrolla la modelación matemática del mismo.

El problema estudiado combina dos importantes problemas de optimización combinatoria: el problema del reparador (también conocido como problema de latencia mínima) y el problema del agente viajero. En ambos problemas se tiene un conjunto de clientes que deben ser visitados solamente una vez por un agente/vehículo que sale de un depósito central y debe regresar a él al finalizar la ruta. La diferencia fundamental entre ambos radica en el criterio de desempeño que se considera en cada caso. En el problema del reparador el objetivo es minimizar la suma de los tiempos de espera de todos los clientes (también conocido como minimizar la latencia total), mientras que en el problema del agente viajero el objetivo es minimizar la distancia recorrida por el vehículo.

En esta tesis se estudia el problema de diseñar una ruta que atienda todos los clientes partiendo de un depósito y regresando a él de tal forma que se minimice tanto el tiempo de espera de cada cliente como la distancia recorrida por el agente. Esto es, estamos en presencia de un problema bi-objetivo.

Para formular matemáticamente el problema bajo estudio, debemos tener en

cuenta los modelos matemáticos que han sido desarrollados para el problema del reparador y para el problema del agente viajero. Estudios previos [1, 2] mostraron que no es conveniente adaptar modelos clásicos basados en el objetivo de distancia para manejar el objetivo de latencia. Es por esto que para modelar nuestro problema se tomará como partida un modelo basado en el objetivo de latencia al cual se le irán incorporando las cuestiones relativas al objetivo de distancia.

Como ya se mencionó anteriormente, existen diversas formulaciones reportadas en la literatura para el problema de latencia mínima, entre las que destancan las formulaciones de Méndez-Díaz et al [33], Gouveia y Voss [20], así como el modelo propuesto por Picard y Queyranne [37] e implementado por Sarubbi [43].

Sin embargo, en el 2012 Angel-Bello et al [1] propusieron dos modelos matemáticos que lograban superar a los anteriores. El modelo que se tomará como base para nuestra formulación es el que los autores denominaron "Modelo A" ya que fue el de mejor desempeño entre los dos que propusieron.

## 3.1 Modelo Matemático

Consideraremos un grafo dirigido y completo $G = (V, A)$, siendo $V = \{0, 1, \ldots, n\}$ el conjunto de los vértices y $A$ el conjunto de los arcos. El subconjunto de vértices $I = \{1, 2, \ldots, n\}$ es el conjunto de los clientes, mientras que el vértice 0 corresponde al depósito o almacén. Cada cliente $i \in I$ tiene asociado un tiempo de servicio $s_i$, y cada arco $(i, j) \in A$ tiene asociado un tiempo de viaje $t_{ij}$, el cual denota el tiempo de recorrido entre el cliente $i$ y el cliente $j$.

De esta forma, se tiene una matriz $C = c_{ij}$ de costos, la cual está conformada de la siguiente manera:

$$c_{ij} = \begin{cases} t_{0j}, & \text{para } i = 0;\ j = 1, 2, \ldots, n; \\ s_i + t_{ij}, & \text{para } i = 1, 2, \ldots, n;\ j = 0, 1, \ldots, n; j \neq i. \end{cases}$$

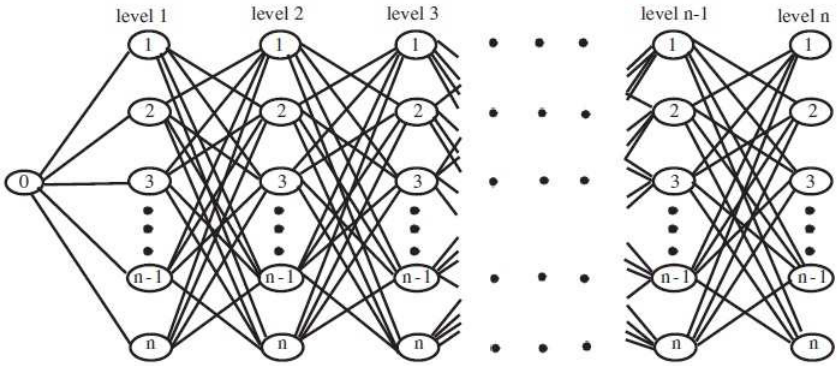



Figura 3.1: Red Multinivel propuesta por Picard y Queyranne.

La formulación del modelo se basa en una red multinivel con $n+1$ niveles (ver figura 3.1), la cual se inspira en la red propuesta en 1978 por Picard y Queyranne [37], para un problema del agente viajero dependiente del tiempo. En esta red, el nivel 0 se refiere al depósito central, o el lugar de donde el agente debe partir y volver al finalizar de visitar a los clientes, mientras que los niveles $1, 2, \ldots, n$, son $n$ copias de los $n$ clientes que el agente debe visitar.

Note que cualquier solución al problema es una permutación de $1, 2, \ldots, n$, a la cual, se le agrega el nodo raíz o depósito en la primera posición. Entonces, cualquier solución puede representarse en esta red siempre que se satisfaga que en cada nivel de la red se escoja solamente un nodo y que este nodo no sea elegido en ningún otro nivel.

Teniendo en cuenta esto, se definen las siguientes variables de decisión:

$$x_i^k = \begin{cases} 1, & \text{Si el nodo } i \text{ es seleccionado en el nivel } k \text{ en la red,} \\ 0, & \text{en caso contrario.} \end{cases}$$

$$y_{ij}^k = \begin{cases} 1, & \text{Si el nodo } i \text{ es seleccionado en el nivel } k \text{ y el nodo } j \text{ en el nivel } k+1 \text{ en la red,} \\ 0, & \text{en caso contrario.} \end{cases}$$

Como hemos comentado, los objetivos a considerar son la latencia total y la distancia total, es decir,

- Objetivo de Latencia. Se desea minimizar el tiempo de espera total de todos los clientes.

$$min\ F_1 = n\sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(n-k)c_{ij}y_{ij}^k$$

- Objetivo de Distancia. Se desea minimizar la distancia total recorrida por el reparador.

$$min\ F_2 = \sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}y_{ij}^k + \sum_{i=1}^{n} c_{i0}x_i^n$$

Note que para la función objetivo de latencia no se considera el arco de regreso al depósito, mientras que para la función objetivo de distancia sí es necesario considerar el costo de ese arco.

Las soluciones factibles deben satisfacer las siguientes restricciones:

- Cada nodo es seleccionado en un solo nivel en la red.

$$\sum_{k=1}^{n} x_i^k = 1,\ i = 1, 2, \ldots, n$$

- En cada nivel en la red se selecciona solamente un nodo.

$$\sum_{i=1}^{n} x_i^k = 1,\ k = 1, 2, \ldots, n$$

- Sólo un arco puede salir del nivel $k$ de la red.

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{ij}^{k} = x_i^k,\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n-1$$

- Sólo un arco puede llegar al nivel $k+1$ en la red

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{ji}^{k} = x_i^{k+1},\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n-1$$

De esta forma, el modelo matemático para el problema que se estudia, es el siguiente:

$$min\ F_1 = n\sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(n-k)c_{ij}y_{ij}^k \tag{3.1}$$

$$min\ F_2 = \sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}y_{ij}^k + \sum_{i=1}^{n} c_{i0}x_i^n \tag{3.2}$$

s.a

$$\sum_{k=1}^{n} x_i^k = 1,\ i = 1, 2, \ldots, n \tag{3.3}$$

$$\sum_{i=1}^{n} x_i^k = 1,\ k = 1, 2, \ldots, n \tag{3.4}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{ij}^{k} = x_i^k,\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n-1 \tag{3.5}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{ji}^{k} = x_i^{k+1},\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n-1 \tag{3.6}$$

$$x_i^k \in \{0, 1\},\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n \tag{3.7}$$

$$y_{ij}^k \geq 0,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n, j \neq i,\ k = 1, 2, \ldots, n \tag{3.8}$$

Note que a pesar de que las variables $y_{ij}^k$ fueron definidas como binarias, en el modelo matemático solo se pide que sean no negativas. Esto se debe a que si tomamos

las restricciones (3.3) y (3.4) y suponemos que $x_r^k = 1$ y $x_t^{k+1} = 1$, tenemos de la restricción (3.5) que $\sum_{l=1,l\neq r}^{n} y_{rl}^k = 1$ y que $\sum_{l=1,l\neq i}^{n} y_{il}^k = 0$ para $i = 1, 2, \ldots, n, i \neq r$. Esto significa, que las variables $y_{lt}^k$ para $j = 1, 2, \ldots, n, j \neq r$ solamente pueden tomar valores positivos.

Por otra parte, de la restricción (3.6), tenemos que $\sum_{l=1,l\neq t}^{n} y_{lt}^k = 1$ y $\sum_{l=1,l\neq j}^{n} y_{lj}^k = 0$ para $j = 1, 2, \ldots, n, j \neq t$. Esto significa que las variables $y_{lt}^k$ para $l = 1, 2, \ldots, n, l \neq t$ solamente pueden tomar valores diferentes de cero.

Estos dos resultados nos dicen que los arcos del nivel $k$ al nivel $k+1$ de la red, deben salir del nodo $r$. Además, el único nodo en el nivel $k+1$ al que puede llegar algún arco es el nodo $t$.

Se tiene también, que de $\sum_{l=1,l\neq r}^{n} y_{rl}^k = 1$, la variable $y_{rt}^k = 1$ mientras que todas las otras variables envueltas en la sumatoria se hacen cero. De forma similar, los arcos que salen del nivel $k$ deben llegar al nodo $t$ en el nivel $k$+1. De $\sum_{l=1,l\neq t}^{n} y_{lt}^k = 1$, la variable $y_{lt}^k = 1$ cuando las variables en la suma son cero.

Como esto se cumple para cualquier nivel, se tiene que $y_{ij}^k \in \{0, 1\}$, $k = 1, 2, \ldots, n-1$, $i, j = 1, 2, \ldots, n; j \neq i$.

## 3.2 Análisis de la complejidad del modelo

La complejidad del modelo propuesto es la siguiente:

- La restricción (3.3), tiene un orden de $n$ restricciones.
- La restricción (3.4), tiene un orden de $n$ restricciones.
- La restricción (3.5), cuenta con $n(n-1)$ restriccciones.

- La restricción (3.6), tiene $n(n-1)$ restriccciones.
  Con lo cual, el modelo completo tiene $2n^2$ restricciones.

- El número de variables binarias $x_i^k$, es $n^2$.

- El número de variables continuas $y_{ij}^k$, es $n^3 - 2n^2 + 2n$.

Si bien con lo explicado anteriormente se logra reducir considerablemente el número de variables binarias, definiendo las $y_{ij}^k$ como continuas, el modelo aún resulta difícil de resolver para instancias con 25 nodos ó más como se verá más adelanteen la sección 3.4 Hay que enfatizar que cada uno de los problemas mono-objetivos que subyacen en el problema bi-objtivo bajo estudio es un problema cominatorio difícil. Particularmente, FULANO DE TAL DEMOSTRÓ EN TAL AÑO QUE EL PROBLEMA DE LATENCIA MÍNIMA ES xxxxxx (REFERENCIA). pOR SU PARTE, EL PROBLEMA DEL AGENTE VIAJERO ES UN PROBLEMA xxxxxx SEGÚN SE DEMOSTRÓ EN XXXXXXD .

## 3.3 Análisis del conflicto entre los objetivos

Después de formular el modelo matemático para el problema biobjetivo bajo estudio, lo siguiente es resolverlo con algún método para verificar su alcance. Sin embargo, ¿cómo justificar el empleo de alguna técnica multiobjetivo? Como ya se comentó anteriormente, la optimización simultánea de todos los objetivos es usualmente imposible, esto es debido al grado de conflicto que por lo general existe entre los objetivos. Es por esto, que el enfoque multiobjetivo en lugar de intentar determinar un óptimo, pretende establecer un conjunto de soluciones eficientes o un frente de Pareto [41].

Por consiguiente, es recomendable verificar primeramente que los objetivos estén realmente en conflicto. Para lo cual, desarrollaremos una prueba de hipótesis

y con este fin denominaremos $F_{1_{opt}}$ al valor óptimo de la función objetivo de latencia y al valor óptimo de la función objetivo de distancia se le nombrará como $F_{2_{opt}}$. Al encontrar la solución óptima para el problema mono objetivo que optimiza $F_1$ esta se evaluará en $F_2$ para determinar qué valor de distancia le corresponde y será denotado como $F_{2_{eval}}$. Así mismo, la solución óptima del problema mono objetivo que optimiza la distancia será evaluada en la función objetivo de latencia y al valor obtenido lo denotaremos como $F_{1_{eval}}$.

Recapitulando,

- $F_{1_{opt}}$, es el valor óptimo en la función objetivo de latencia.
- $F_{1_{eval}}$, es la evaluación del óptimo de distancia en la función objetivo de latencia.
- $F_{2_{opt}}$, es el valor óptimo en la función objetivo de distancia.
- $F_{2_{eval}}$, es la evaluación del óptimo de latencia en la función objetivo de distancia.

Con lo anterior, la hipótesis nula se define como: la media ($\mu$) del óptimo en la función de latencia ($F_{1_{opt}}$) es igual a la media de la diferencia entre la evaluación del óptimo de latencia en la función objetivo de distancia ($F_{2_{eval}}$) y el óptimo de la función objetivo de distancia ($F_{2_{opt}}$). La hipótesis alternativa queda definida como que la media del óptimo en la función de latencia ($F_{1_{opt}}$) no es igual a la media de la diferencia entre la evaluación del óptimo de latencia en la función objetivo de distancia ($F_{2_{eval}}$) y el óptimo de la función objetivo de distancia ($F_{2_{opt}}$).

$$H_0 : \mu_{F_{1_{opt}}} = \mu_{F_{2_{eval}} - F_{2_{opt}}}$$

$$H_1 : \mu_{F_{1_{opt}}} \neq \mu_{F_{2_{eval}} - F_{2_{opt}}}$$

Es decir, se aceptará la hipótesis nula si el óptimo en la función objetivo de latencia es igual a la evaluación del óptimo de la función objetivo de latencia en la función objetivo de distancia, menos el óptimo de la función objetivo de distancia.

Para las hipótesis nula y alternativa del objetivo de distancia, es similar. La hipótesis nula queda definida como que el óptimo en la función de distancia ($F_{2_{opt}}$) es igual a la diferencia entre la evaluación del óptimo de distancia en la función objetivo de latencia ($F_{1_{eval}}$) y el óptimo de la función objetivo de latencia ($F_{1_{opt}}$). La hipótesis alternativa queda definida como que esta diferencia no es igual.

$$H_0 : \mu_{F_{2_{opt}}} = \mu_{F_{1_{eval}} - F_{1_{opt}}}$$

$$H_1 : \mu_{F_{2_{opt}}} \neq \mu_{F_{1_{eval}} - F_{1_{opt}}}$$

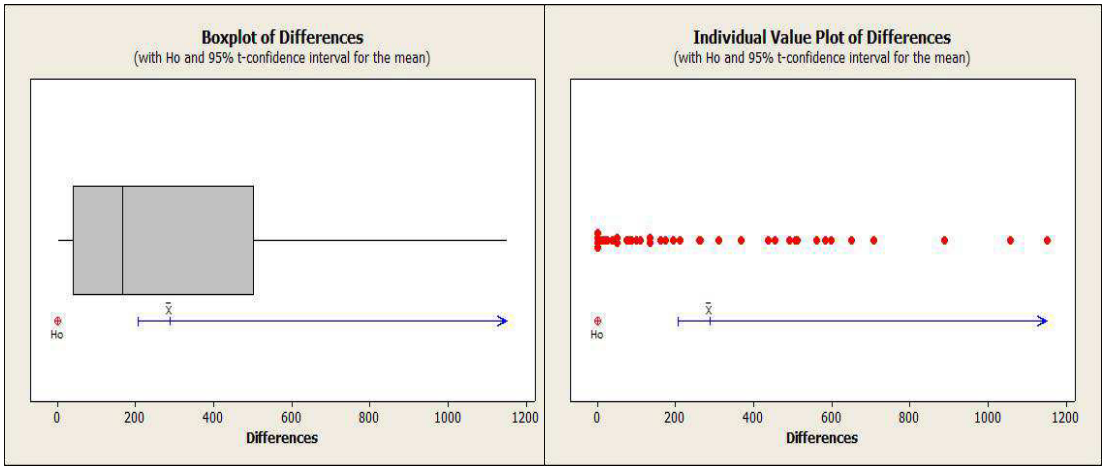
Para hacer la prueba de hipótesis se usó un subconjunto de las instancias que serán utilizadas posteriormente para los experimentos computacionales y que se describen en el capítulo 5. Específicamente aquí usamos 20 instancias de 10 clientes y 20 instancias de 20 clientes de las instancias denominadas TRP-S que fueron generadas por Salehipour et al [42] para resolver el problema del reparador. Cada problema mono objetivo en cada instancia fue resuelto en CPLEX v.11.2 en un servidor Sun Fire V440 con 4 procesadores Ultra Sparc III a 1062 GHZ con 8 Gb de RAM

Después de tomar los resultados, se ingresaron al software Design Expert y la prueba de hipótesis se realizó con un $\alpha$=0.05, es decir, con una confiabilidad del 95 %. El software reportó lo siguiente para el objetivo de latencia:

95 % lower bound for mean difference: 205.7

T-Test of mean difference $= 0 (vs > 0)$ : T-Value = 5.95 P-Value = 0.000

| | N | Mean | StDev | SE Mean |
|---|---|---|---|---|
| LatEval | 40 | 2518 | 1159 | 183 |
| LatenciaOpt | 40 | 2231 | 1004 | 159 |
| Difference | 40 | 286.9 | 304.8 | 48.2 |




Con los resultados obtenidos en Design Expert, podemos utilizar la prueba del valor $P$. Este valor $P$ es la probabilidad de que el estadístico de prueba asuma un valor que sea al menos tan extremo como el valor observado del estadístico cuando la hipótesis nula es verdadera. En términos más formales, el valor $P$ se define como el nivel de significación menor que llevaría a rechazar la hipótesis nula.

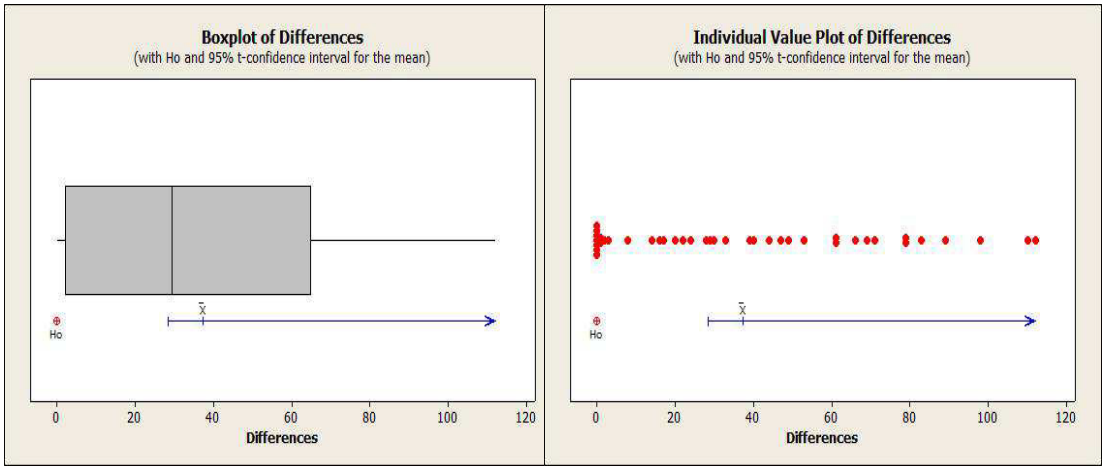
Este valor $P$ nos lo brinda el software, y como puede observarse es menor que el valor de significancia $\alpha$. Con lo anterior, y utilizando que si el valor de $P$ es menor que el valor del $\alpha$ se rechaza la hipótesis nula, entonces podemos aceptar la hipótesis alternativa que nos dice que la diferencia entre el óptimo de latencia no es igual a la diferencia entre la distancia evaluada y la distancia óptima.

De forma similar, se realizó el análisis para el objetivo de distancia.

95 % lower bound for mean difference: 28.36

| | N | Mean | StDev | SE Mean |
|---|---|---|---|---|
| DistanciaEval | 40 | 373.7 | 68.3 | 10.8 |
| DistOpt | 40 | 336.3 | 48.7 | 7.7 |
| Difference | 40 | 37.45 | 34.12 | 5.40 |

T-Test of mean difference $= 0 (vs > 0)$ : T-Value $= 6.94$ P-Value $= 0.000$




Con lo cual, se vuelve a rechazar la hipótesis nula porque nuevamente el valor de $P$ es menor que el valor del $\alpha$. Podemos entonces, aceptar la hipótesis alternativa que nos dice que la diferencia entre el óptimo de distancia no es igual a la diferencia entre la latencia evaluada y la latencia óptima.

De esta forma, corroboramos que los objetivos están en conflicto y se justifica la necesidad de utilizar una técnica multiobjetivo para resolver el modelo.

## 3.4 Alcance del modelo

En esta sección se muestra el tamaño de las instancias que pueden ser resueltas a optimalidad utilizando el modelo propuesto, para lo cual, se utilizará el método de

la $\epsilon$-restricción.

En un problema biobjetivo este método consiste en optimizar sólo un objetivo considerando al otro objetivo como una restricción acotada por un cierto valor $\epsilon$. Es un proceso iterativo, donde el $\epsilon$ se actualiza iteración a iteración para que recorra todo el intervalo desde el punto anti-ideal hasta el punto ideal. Esto se realiza con un "salto" en el épsilon el cual es definido por el usuario.

Primeramente se resuelven los dos problemas mono objetivo independientemente, esto es,

$$min\ F_1 = n\sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(n-k)c_{ij}y_{ij}^k \tag{3.9}$$

s.a

las restricciones (3.3) a (3.8)

y

$$min\ F_2 = \sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}y_{ij}^k + \sum_{i=1}^{n} c_{i0}x_i^n \tag{3.10}$$

s.a

las restricciones (3.3) a (3.8)

El óptimo (3.9) se guarda en $F_{1_{opt}}$. El óptimo de (3.10) se guarda en $F_{2_{opt}}$. Cada una de las soluciones óptimas se evalúan en la otra función objetivo, es decir, la solución correspondiente a $F_{1_{opt}}$ se evalúa en la función objetivo de distancia y se obtiene $F_{2_{eval}}$, mientras que la solución óptima correspondiente a $F_{2_{opt}}$ se evalúa en la función objetivo de latencia y se obtiene $F_{1_{eval}}$

Teniendo estos valores, ya conocemos el punto ideal y anti-ideal de la instancia,

es decir, los mejores valores que se pueden obtener y los peores que se pueden obtener.

$$(F_{1_{opt}}, F_{2_{opt}}),\ (F_{2_{eval}}, F_{1_{eval}})$$

Teniendo estos valores, se decidió aplicar el método de la $\epsilon$-restricción resolviendo sobre la función objetivo de distancia, esto debido a que experimentos preliminares mostraron que la función objetivo de latencia es más difícil de resolver. Así que la función objetivo de latencia se convertirá en la restricción acotada por el épsilon. Para esto, definimos el rango de variación del épsilon en el siguiente intervalo.

$$F_{1_{eval}} \geq \epsilon \geq F_{1_{opt}}$$

Es decir, el valor del épsilon solo fluctuará entre el peor valor que puede tener la latencia ($F_{1_{eval}}$) y el mejor que se puede obtener ($F_{1_{opt}}$). El modelo para el método de la $\epsilon$-restricción, queda de la siguiente manera:

$$min\ F_2 = \sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}y_{ij}^k + \sum_{i=1}^{n} c_{i0}x_i^n \tag{3.11}$$

s.a

$$n\sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n} (n-k)c_{ij}y_{ij}^k \leq \epsilon \tag{3.12}$$

$$\sum_{k=1}^{n} x_i^k = 1,\ i = 1, 2, \ldots, n \tag{3.13}$$

$$\sum_{i=1}^{n} x_i^k = 1,\ k = 1, 2, \ldots, n \tag{3.14}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{ij}^k = x_i^k,\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n-1 \tag{3.15}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{ji}^k = x_i^{k+1},\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n-1 \tag{3.16}$$

$$x_i^k \in \{0, 1\},\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n \tag{3.17}$$

$$y_{ij}^{k} \geq 0,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n, j \neq i,\ k = 1, 2, \ldots, n \qquad (3.18)$$

En nuestra implementación del método de la $\epsilon$-restricción se realiza un salto de una unidad en el épsilon, hasta llegar al punto $(F_{1_{opt}}, F_{2_{eval}})$.

Las instancias utilizadas para medir el alcance del modelo propuesto son las instancias reportadas en el trabajo de Salehipour et al [42], así como las instancias reportadas por Angel-Bello et al [1].

El modelo (3.11)-(3.18), fue resuelto en CPLEX v.11.2 en un servidor Sun Fire V440 con 4 procesadores Ultra Sparc III a 1062 GHZ con 8 Gb de RAM.

Debido a la complejidad del problema, no fue posible en todos los casos encontrar los frentes de Pareto óptimos, por lo que se le dio un límite de tiempo de dos horas a cada una de las instancias de ambos conjuntos de instancias de prueba. Con esta restricción de tiempo, se logró resolver instancias de hasta 20 clientes, en ambos conjuntos de instancias utilizadas. Es decir, el modelo propuesto no puede ser resuelto a optimalidad para 30 o más nodos.

En el capítulo de experimentación computacional, se dedica una sección para mostrar los frentes de Pareto encontrados de forma exacta con el método de la $\epsilon$-restricción para algunas instancias reportadas en el trabajo de Angel-Bello et al [1] y para algunas instancias reportadas en el trabajo de Salehipour et al [42]. Estos frentes de Pareto óptimos también serán usados para comparar los resultados que se obtengan con el método metaheurístico desarrollado.

Capítulo 4

# Método de solución

Como ya fue mencionado anteriormente, el problema que se estudia es un problema biobjetivo que tiene como criterios el minimizar tanto la distancia como la latencia de la ruta. Como se ilustró en el capítulo anterior, el modelo propuesto no puede ser resuelto a optimalidad para 30 o más nodos. Por esto, se decidió utilizar una metaheurística para resolver el problema.

## 4.1 NSGA-II

Así como puede obtenerse el frente exacto de Pareto por medio de un algoritmo iterativo, puede buscarse una aproximación al frente de Pareto por medio de algún método heurístico o metaheurístico, es decir, por medio de algún método que brinde buenas soluciones cercanas al frente exacto pero con un menor tiempo de cómputo. En nuestro caso, se optó por seguir el algoritmo del NSGA-II propuesto en el 2000 por Debb et al [17].

Esta metaheurística es un algoritmo genético elitista que es muy utilizado en los problemas multiobjetivo gracias a los buenos resultados que ofrece. El pseudo-código general del NSGA-II se describe a continuación.

**Algorithm 1** NSGA-II

1: t=0
2: Generar población inicial $P_0$ de tamaño *n_pob*.
3: Generar población de hijos, $Q_0$ de tamaño *n_pob* a partir de $P_0$ por medio de selección, combinación y mutación.
4: $R_0 = P_0 \bigcup Q_0$.
5: **repeat**
6: Divide $R_t$ en frentes de no dominancia $F_i$.
7: $P_{t+1} = 0$ e $i = i + 1$.
8: **while** $|P_{t+1}|+|F_i| < n\_pob$ **do**
9: i=i+1.
10: $P_{t+1}$=$P_{t+1} \bigcup F_i$
11: i=i+1;
12: **end while**
13: Ordenar los elementos de los frentes por su cwd_dst en orden decreciente.
14: Incluir los *n_pob* elementos posibles de los frentes $F_i$ a $P_{t+1}$.
15: Generar la población de hijos $Q_{t+1}$ de tamaño *n_pob* a partir de $P_{t+1}$ por medio de selección, combinación y mutación.
16: $R_{t+1} = P_{t+1} \bigcup Q_{t+1}$
17: t=t+1
18: **until** criterio de paro.

Este algoritmo opera sobre una población de soluciones, y utiliza operadores típicos de los algoritmos genéticos tales como selección de padres, cruce, mutación, selección de individuos que pasan a la siguiente generación, etcétera. El proceso completo se repite hasta que se alcanza un criterio de paro, que en nuestro caso es un número prefijado de iteraciones o generaciones. A continuación, se detallarán los componentes de la implementación desarrollada para resolver el problema que se está estudiando.

## 4.2 Generación de la población inicial $P_0$

Como se dijo anteriormente, el algoritmo del NSGA-II opera sobre un conjunto de soluciones.

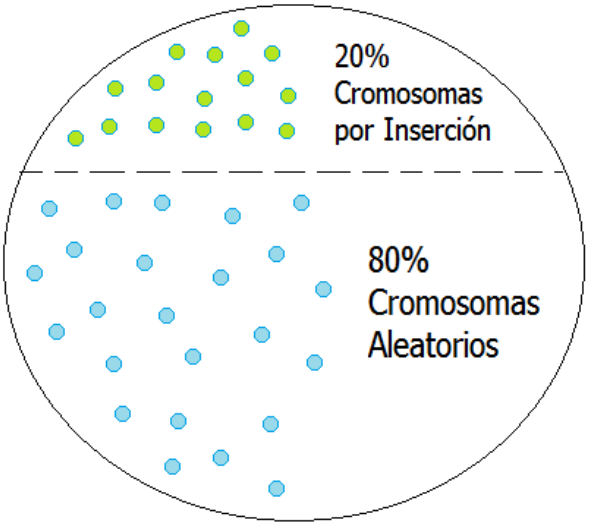



Figura 4.1: Generación de la población inicial

En nuestra implementación, generaremos la población inicial (paso 2 del algoritmo) de la siguiente manera (ver figura 4.1).

- 80 % de las soluciones de $P_0$ se generarán de forma aleatoria.

- 20 % de las soluciones de $P_0$ se generarán utilizando la heurística de inserción más barata. Una breve explicación de esta heurística puede encontrarse en la sección 1.1.1 de este trabajo.

Con el fin de asegurar que la población inicial está compuesta totalmente por individuos diferentes, cada vez que se genera una solución, bien sea aleatoriamente o mediante la heurística de inserción, se comprueba que sea diferente de las que ya se encuentran en $P_0$ antes de agregarla a la población.

## 4.3 Selección de los padres

Una vez que tenemos una población inicial, se seleccionan las soluciones que serán combinadas para formar la población de hijos $Q$ (paso 3 del algoritmo). Estas soluciones serán seleccionadas por medio de torneos binarios.

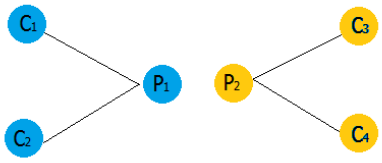



Figura 4.2: Torneo Binario

En la figura 4.2, se aprecia de forma gráfica el funcionamiento de un torneo binario, el cual consiste en seleccionar dos individuos de forma aleatoria y elegir el mejor de ellos. Este proceso se realizó en la selección de ambos padres, quienes para diferenciarlos, serán renombrados como $P_1$ y $P_2$.

La selección de la primer solución $P_1$ es libre entre todos los cromosomas que

pertenecen a la población actual, sin embargo, la selección de $P_2$ depende del que ha sido seleccionado previamente para así evitar que se seleccione el mismo cromosoma. Cada par de padres seleccionados se combinan para generar nuevas soluciones hijas como se explicará más adelante.

Este proceso de seleccionar padres para combinarlos, se repite hasta que la matriz de hijos $Q$ esté completa.

## 4.4 Combinación. Generación de la población $Q_0$.

El proceso de combinación o cruzamiento de soluciones para formar a $Q_0$, está basado en el cruzamiento OX descrito en el artículo de Oliver et al [35] y publicado en 1987. Este cruzamiento se describe a continuación y se ilustra gráficamente en la figura 4.3.

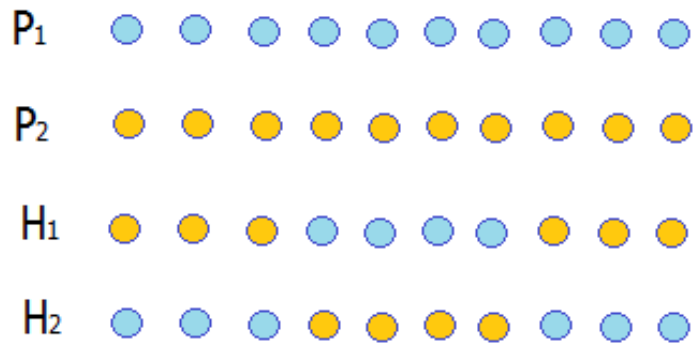



Figura 4.3: Cruzamiento OX.

Supongamos que las soluciones $P_1$ y $P_2$ fueron seleccionados de la población

inicial como padres y son los siguientes.

$$P_1 = 1\ 3\ 2\ 6\ 4\ 5\ 9\ 7\ 8$$

$$P_2 = 3\ 7\ 8\ 1\ 4\ 9\ 2\ 5\ 6$$

Para realizar la combinación o cruzamiento, se seleccionan aleatoriamente dos posiciones de corte $i$ y $j$, asegurando que se cumple siempre que $i < j$. La subcadena del padre $P_1[i]$ a $P_1[j]$ se copia al primer hijo $H_1$, desde $H_1[i]$ a $H_1[j]$. Sean por ejemplo,

$$i = 4,\ j = 6$$

$$P_1 = 1\ 3\ 2\ 6\ 4\ 5\ 9\ 7\ 8$$

$$P_1 = \_\ \_\ \_\ 6\ 4\ 5\ \_\ \_\ \_$$

Finalmente, el primer hijo $H_1$ se completa con el padre $P_2$ de forma circular a partir de $i - 1$ en adelante, ubicando los nodos faltantes en él hasta completarlo.

$$P_2 = 3\ 7\ 8\ 1\ 4\ 9\ 2\ 5\ 6$$

$$H_1 = 8\ 1\ 9\ 6\ 4\ 5\ 2\ 3\ 7$$

El segundo hijo $H_2$ se obtiene de la misma forma, seleccionando la trayectoria de $i$ a $j$ en el padre $P_2$, y completándolo con el padre $P_1$.

$$P_2 = 3\ 7\ 8\ 1\ 4\ 9\ 2\ 5\ 6$$

$$H_2 = \_\ \_\ \_\ 1\ 4\ 9\ \_\ \_\ \_$$

$P_1$= 1 3 2 6 4 5 9 7 8

$H_2$= 2 6 5 1 4 9 7 8 3

La selección de padres y su combinación, se realizan un número pre-establecido de veces, hasta alcanzar el tamaño deseado en la población de hijos $Q_0$ .

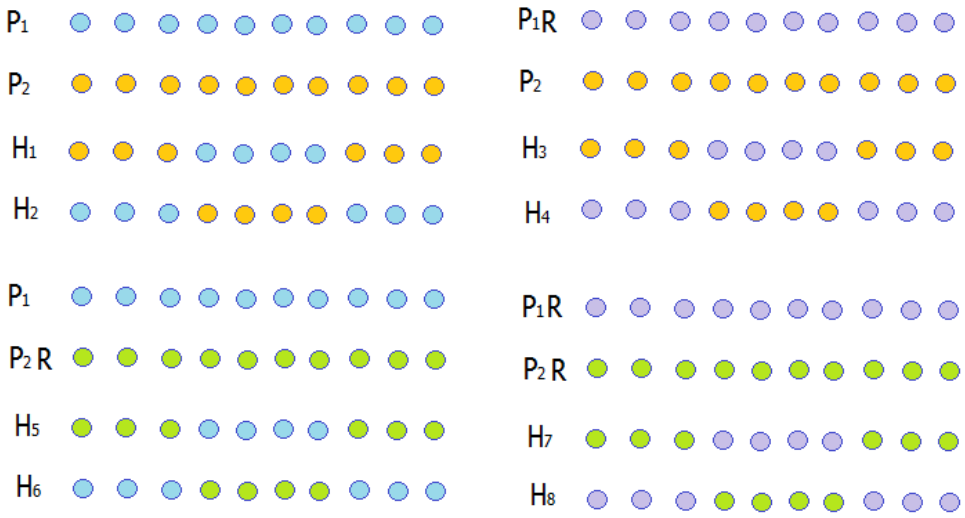



Figura 4.4: Cruza donde se obtienen 8 hijos

Note que en la obtención de los hijos mediane el cruzamiento OX, influye grandemente el orden de los nodos en la permutación. Si este orden se cambia en alguno de los padres, los hijos obtenidos serán diferentes. Con el propósito de obtener mayor diversidad, Prins [34] aplica el cruzamiento OX invirtiendo el orden en cada uno de los padres separadamente y luego simultáneamente para así obtener ocho hijos.

El procedimiento de cruza donde se obtienen ocho hijos se puede observar gráficamente en la figura 4.4. Este proceso está descrito en [34] y se deriva directamente del procedimiento OX de Oliver et al [35], anteriormente explicado.

Para ilustrar, sean $P_1$ y $P_2$ las mismas soluciones seleccionadas previamente

$$P_1 = 1\ 3\ 2\ 6\ 4\ 5\ 9\ 7\ 8$$

$$P_2 = 3\ 7\ 8\ 1\ 4\ 9\ 2\ 5\ 6$$

Sus versiones invertidas, serían las siguientes:

$$P_1R = 8\ 7\ 9\ 5\ 4\ 6\ 2\ 3\ 1$$

$$P_2R = 6\ 5\ 2\ 9\ 4\ 1\ 8\ 7\ 3$$

Es así, como cuatro cruzamientos del tipo OX se llevan a cabo entre las cuatro combinaciones posibles de padres: $(P_1, P_2)$, $(P_1R, P_2)$, $(P_1, P_2R)$ y $(P_1R, P_2R)$. Cada una de estas parejas nos dará dos hijos, y en total, obtendremos ocho hijos por cruza.

Ambas aplicaciones del operador OX fueron implementadas y validadas computacionalmente de forma independiente, esto es, utilizando una u otra forma de cruzamiento. En el capítulo siguiente se describen los resultados de los experimentos realizados.

## 4.5 Mutación

En un algoritmo genético, la mutación está diseñada para agregarle diversidad a la población y así ampliar la posibilidad de explorar el espacio de búsqueda completo [44].

En nuestra implementación del NSGA-II, se eligió como mutación una búsqueda local basada en movimientos 2-opt [15]. Este movimiento consiste en eliminar dos arcos y reconectar los dos caminos resultantes de una manera diferente

para obtener una nueva solución. Cabe señalar, que existe una única forma de reconectar el tour. El conjunto de soluciones que se pueden obtener de esta forma, a partir de una solución dada, constituye el vecindario o entorno a explorar.

La búsqueda local comienza a partir de una solución y busca mejorarla progresivamente, es decir, se basa en explorar el entorno de una solución y seleccionar una nueva solución en él la cual mejora el valor de la función objetivo que guía la búsqueda. Desde la nueva solución se explora su entorno y se repite el proceso.

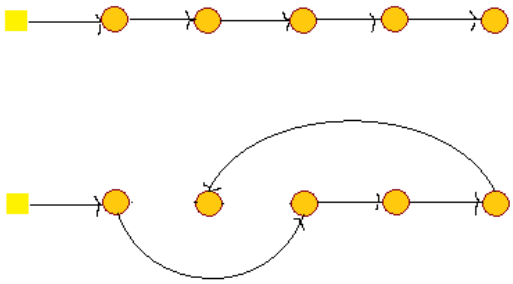


Figura 4.5: Movimiento 2-Opt

En nuestra implementación, la función que guía esta búsqueda es la latencia de la ruta y cada cromosoma tiene un 15 % de probabilidad de mutación.

## 4.6 Generación de la población $R_0$

Esta población es la unión entre la población inicial $P_0$ y la población de hijos $Q_0$ (paso 4 del algoritmo), como se muestra en la figura 4.6.

Es importante considerar todas las soluciones generadas hasta el momento para poder seleccionar las que se conservarán para la siguiente generación en la población. Esto se lleva a cabo mediante una separación en capas de no dominancia

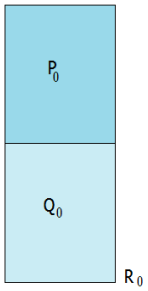



Figura 4.6: Matriz $R_0$

que se explica a continuación.

## 4.7 Separación de $R_0$ en capas de no dominancia

Dado un conjunto de soluciones $R_0$, este conjunto se puede descomponer en capas de no-dominancia, $F_1, F_2, \ldots$, de la siguiente manera (paso 6 del algoritmo):

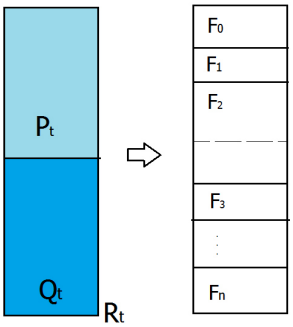



Figura 4.7: Capas de no dominancia en el NSGA-II

- $F_1$: Conjunto de soluciones no dominadas de $R_0$.
- $F_2$: Conjunto de soluciones no dominadas de $R_0 \backslash F_1$.
- $F_3$: Conjunto de soluciones no dominadas de $R_0 \backslash \{F_1 \bigcup F_2\}$

y así sucesivamente. Esto puede ser observado en la imagen 4.7.

Para determinar los frentes de no dominancia en nuestro algoritmo, se utilizó el procedimiento descrito en Debb et al [17], denominado *Fast non-dominated sort.* Para lograrlo, a cada solución se le calcularon dos valores:

- $n_p$, el número de soluciones que dominan a la solución $p$.
- $S_p$, el conjunto de soluciones que $p$ domina.

El primer frente de no dominancia es formado con aquellas soluciones para las cuales $n_p = 0$. De ahí, se investigan las soluciones que pertenecen a su conjunto $S_p$ y a cada una de las soluciones que pertenezcan a él, se les resta en una unidad su contador $n_p$. Las soluciones con un $n_p = 0$ formarán el segundo frente y así sucesivamente hasta que todas las soluciones hayan sido rankeadas.

En el peor de los casos, para cada solución $p$ a partir del segundo nivel tendrán un contador a lo más de $n - 1$, lo que hará que sea visitada a lo más $n - 1$ veces para poder ser rankeada. Por lo tanto, la complejidad total de este rankeo tiene un orden de $n^2$ [17].

## 4.8 Selección de la siguiente generación

Los pasos 7 al 11 del algoritmo se refieren a la selección de las soluciones que pasarán a formar parte de la siguiente generación esto es, de la generación con la que se iniciará nuevamene todo el proceso a partir del paso 3.

Para ello, se puede asociar a cada solución una medida de dispersión de los puntos no dominados [17], para lo cual se define lo siguiente para cada solución:

- $cwd\_dst(P_i)$: medida de la dispersión. Este número mide en cierta forma la dispersión de los puntos no dominados aproximando el perímetro del cuadrado

formado con los puntos no dominados más cercanos a la solución $P_i$ que se está investigando. Esta medida puede observarse gráficamente en la imagen 4.8

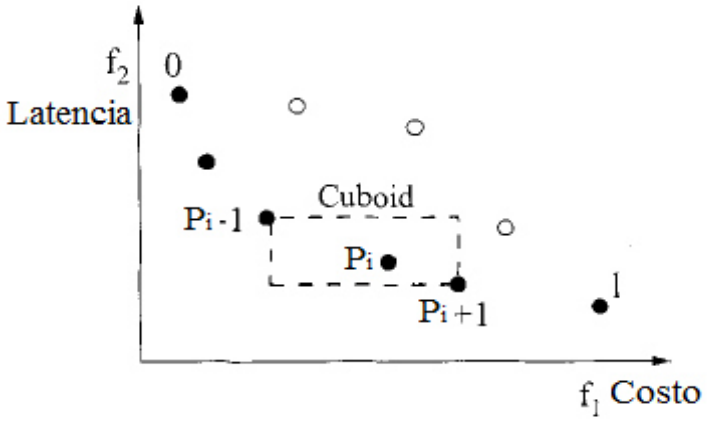



Figura 4.8: Medida de dispersión en el NSGA-II

Para formar la nueva generación $P_1$ se van incluyendo los frentes de no dominancia $F_0, F_1, \ldots$ de $R_0$ completamente mientras no se sobrepase el tamaño de población deseado. Si algún frente $F_i$ no cabe completo, se introducen las soluciones que tengan la medida de dispersión más alta en él, esto es para asegurar la diversidad de la población. Esta población $P_1$, debe tener el tamaño de la población inicial, y es la población que será considerada como nuestra nueva población inicial.

En el capítulo de experimentación computacional, se dedica una sección para mostrar los frentes encontrados por nuestro NSGA-II, utilizando las instancias reportadas en el trabajo de Angel-Bello et al [1] y las instancias reportadas en el trabajo de Salehipour et al [42].

Capítulo 5

# Experimentación Computacional

Las metodologías expuestas anteriormente, el método de la $\epsilon$-restricción y la metaheurística NSGA-II, fueron implementadas computacionalmente. En este capítulo se presentan los frentes de Pareto exactos obtenidos con el método de la $\epsilon$-restricción, así como los frentes aproximados obtenidos con nuestra implementación del NSGA-II.

## 5.1 Descripción de las instancias

Para la realización de los experimentos computacionales, utilizamos dos conjuntos de instancias con características similares.

- Instancias denominadas GTRP-$S_0$ [1], las cuales fueron generadas aleatoriamente, mediante puntos con coordenadas reales a partir de una distribución uniforme definida entre 0 y 100. En estas instancias, las distancias euclídeas se redondean a enteras y serán consideradas como los tiempos de viaje $t_{ij}$. Los tiempos de servicio $s_i$ se consideraron nulos.

  Se dispone de diferentes tamaños de instancias respecto al número de clientes: 10, 15, 20, 25, 30, 35, 40, 60, 80 y 100 clientes. Cada grupo cuenta con 25 instancias, para un total de 250 instancias.

- Instancias denominadas TRP-S [42], las cuales se dividen en siete grupos, desde

10 hasta 1000 vértices: 10, 20, 50, 100, 200, 500 y 1000 vértices. Se tienen veinte instancas aleatorias para cada uno de los siete grupos anteriores, para un total de 140 instancias.

Las coordenadas de los vértices fueron generados a partir de una distribución uniforme entre 0 y 100 para las instancias menores a 500 vértices, y generados a partir de una distribución uniforme entre 0 y 500 para las instancias de tamaños 500 y 1000. Todas las distancias utilizadas son euclídeas, redondeadas al entero menor más cercano. Los tiempos de servicio son todos nulos.

## 5.2 Métricas de desempeño

En optimización multiobjetivo existen diversas métricas que permiten evaluar la calidad del frente obtenido. Para diseñar una métrica eficiente para una comparación de métodos multiobjetivo se suelen considerar tres cuestiones.

1. Minimizar la distancia entre el frente de Pareto aproximado, el cual fue producido por el algoritmo que se propone, respecto al frente de Pareto verdadero. Esto, suponiendo que se conoce el frente exacto.

2. Maximizar la distribución de las soluciones encontradas. De esta forma, se busca obtener una distribución de soluciones no dominadas lo más suave y uniforme posible.

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

Sin embargo, una métrica que satisfaga con un solo valor númerico los tres elementos anteriores es inexistente [51]. Por lo anterior, se recomienda utilizar más de una métrica para evaluar el desempeño de un algoritmo. En nuestro caso, se utilizarán las siguientes métricas:

1. **Número de puntos en el frente**. Esto se refiere a la capacidad del algoritmo propuesto de encontrar puntos eficientes. Esta es una medida importante debido a que las fronteras eficientes que proveen más alternativas son preferibles ante las fronteras que proveen menos puntos.

2. **Tamaño del espacio cubierto**. Esta métrica fue propuesta por Zitzler y Thiele en 1999 [51] y fue llamada "Size of the Space Covered" ó SCC por sus siglas en inglés. Esta métrica estima el volumen de los puntos dominados, por lo tanto, entre mayor sea este valor, mejor.

3. **k-Distancia**. Esta métrica fue propuesta por Zitzler, Laumanns y Thiele en el 2001 [50] y es una técnica para estimar la densidad del frente. Esta métrica mide la distancia al k-ésimo punto eficiente más cercano. Este valor, entre más pequeño sea, mejor ya que entre más cercanos estén los puntos, más denso es el frente.

4. **Métrica** $M_1^*$. Propuesta por Zitzler en su tesis doctoral [49]. Esta métrica consiste en calcular la distancia euclideana promedio entre la frontera de Pareto aproximada y el frente exacto. En este caso, entre menor sea la distancia, mejor es la aproximación al frente exacto.

5. **Medida de cobertura entre dos conjuntos.** Esta métrica fue propuesta por Zitzler y Thiele en 1999 [51] y con ella se comparan dos conjuntos de soluciones no dominadas, supongamos $A$ y $B$, de forma que el valor $C(A, B)$ permite, mediante la fórmula

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

calcular el porcentaje de soluciones de $B$ que son débilmente dominadas por al menos una solución de $A$. Cuando $C(A, B) = 1$, esto indica que todas las soluciones de $B$ son débilmente dominadas por las de $A$, al contrario, si $C(A, B) = 0$, ninguna solución de $B$ es débilmente dominada por las soluciones de $A$.

Sin embargo, $C(A, B)$ no necesariamente es igual a $1 - C(A, B)$, por lo que se deben calcular $C(A, B)$ y $C(B, A)$ siempre.

## 5.3 Frentes de pareto exactos

El problema que se quiere resolver, tal como está planteado en el capítulo de descripción y modelación del problema, consiste en minimizar dos objetivos: latencia y distancia.

Como se comentó anteriormente utilizaremos el método de la $\epsilon$-restricción para encontrar los frentes exactos. En nuestro caso, el modelo mono-objetivo que se resuelve con el método de la $\epsilon$-restricción tiene la función objetivo de distancia mientras que la función objetivo de latencia es considerada como una restricción. Esto se decidió así porque los problemas con el objetivo de latencia son más difíciles de resolver desde el punto de vista computacional.

La variación del $\epsilon$ será en el rango

$$\left( F_{1_{eval}}, \quad F_{1_{opt}} \right)$$

con un salto que es determinado por el usuario. En nuestro caso, este salto tiene el valor de uno.

Para obtener los frentes de Pareto exactos se utilizaron 75 instancias GTRP-$S_0$ (25 de 10 clientes, 25 de 15 clientes y 25 de 20 clientes) y 40 instancias TRP-S (20 de 10 clientes y 20 de 20 clientes). Se utilizó CPLEX v.11.2 en un servidor Sun Fire V440 con 4 procesadores Ultra Sparc III a 1062 GHZ con 8 Gb de RAM.

A continuación se muestran los frentes de Pareto obtenidos para algunas instancias. En todas las gráficas, el eje de coordenadas X representa el objetivo de distancia mientras que el eje de las coordenadas Y representa el objetivo de latencia.

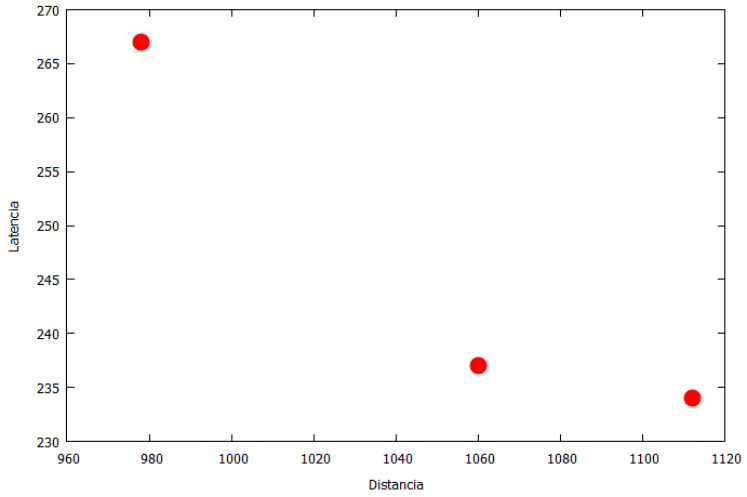



Figura 5.1: Frente de pareto exacto para la instancia TRP-S10-05

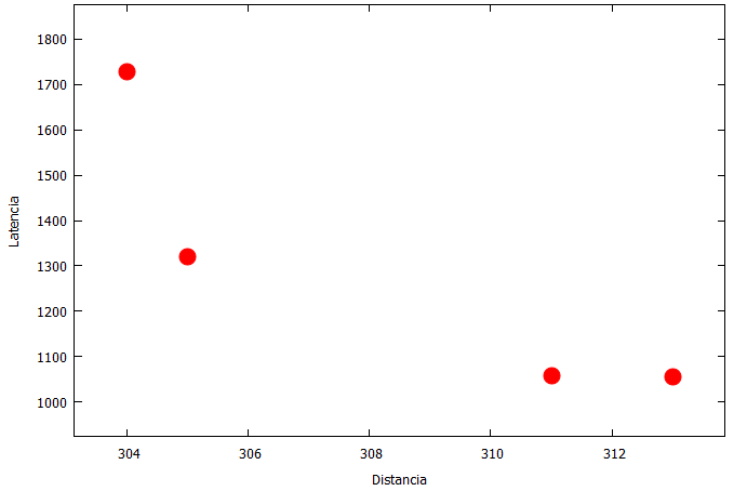



Figura 5.2: Frente de pareto exacto para la instancia GTRP-$S_0$10-09

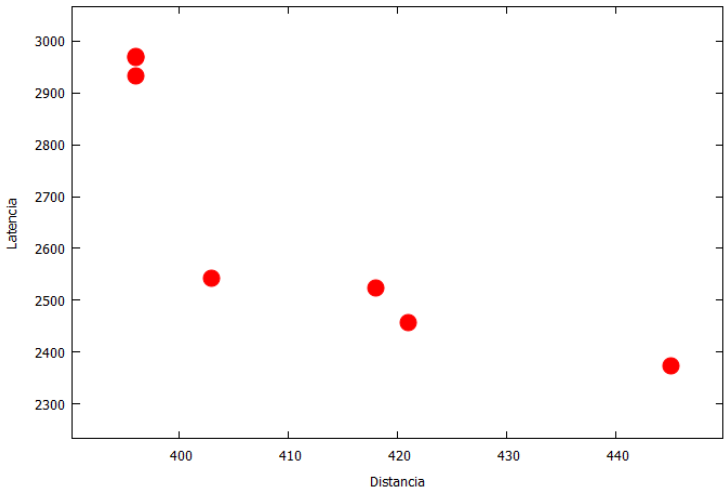



Figura 5.3: Frente de pareto exacto para la instancia GTRP-$S_0$15-16

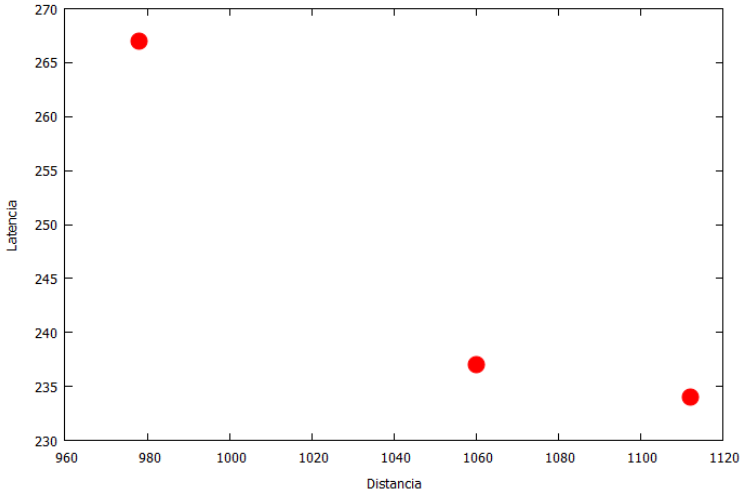



Figura 5.4: Frente de pareto exacto para la instancia TRP-S20-20

En la tabla 5.1 se puede observar el promedio de tiempo computacional que se tarda el algoritmo de la $\epsilon$-restricción en cada grupo de instancias. Es evidente, que conforme crece el número de clientes, crece muchísimo el tiempo computacional.

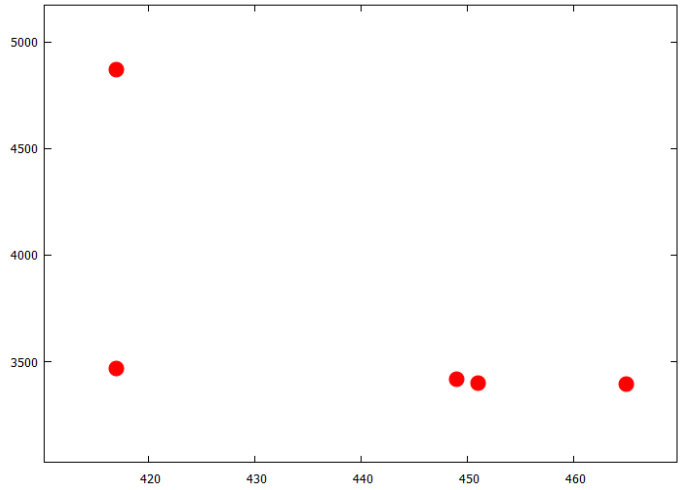



Figura 5.5: Frente de pareto exacto para la instancia GTRP-$S_0$20-15

| Instancia | n | # Puntos | k-distancia | CPU time |
|---|---|---|---|---|
| GTRP-$S_0$ | 10 | 3.208333333 | 0.76736325 | 121.0083432 |
| | 15 | 4.782608696 | 0.548146435 | 83296.22799 |
| | 20 | 5.894736842 | 0.205913242 | 283032.672 |
| TRP-S | 10 | 2.9 | 0.581653525 | 101.279375 |
| | 20 | 6.470588235 | 0.225294118 | 150937.6329 |

Tabla 5.1: Métricas de desempeño para los frentes obtenidos en el método de la $\epsilon$-restricción.

Note que el número de puntos obtenidos por el método del $\epsilon$-constraint es reducido, esto debido a que el método solamente encuentra los puntos soportados. Inicialmente, se puede tener un estimado máximo de cuantos puntos nos dará el método, esto es porque el número de veces que se ejecuta el algoritmo depende del tamaño del salto del $\epsilon$. Aún así, no es seguro que en el $\epsilon$ actual se encuentre una solución que satisfaga el modelo, así que este número puede reducirse.

## 5.4 Ajuste de parámetros para nuestra implementación del NSGA-II

Debido a la naturaleza del algoritmo NSGA-II, se tienen tres parámetros a ajustar:

1. Tamaño de la población, en el que se probaron los valores de 100 y 200 soluciones en la población inicial.
2. Probabilidad de mutación, en la que se probó una probabilidad de 15 % y 20 %.
3. Número de iteraciones, en el que se probaron 300 y 500 iteraciones.

Se decidió realizar un diseño factorial $2^3$ en el cual se utilizaron las siguientes instancias de prueba, las cuales fueron elegidas aleatoriamente entre todos los grupos de instancias disponibles.

| | n | Instancia |
|---|---|---|
| GTRP-$S_0$ | 10 | GTRP10s0-08 |
| | | GTRP10s0-11 |
| | 15 | GTRP15s0-08 |
| | | GTRP15s0-17 |
| | 20 | GTRP20s0-02 |
| | | GTRP20s0-12 |
| TRP-S | 10 | TRP-S10-11 |
| | | TRP-S10-15 |
| | 20 | TRP-S20-06 |
| | | TRP-S20-07 |

Tabla 5.2: Instancias elegidas para el ajuste de parámetros.

Para cada instancia elegida se realizaron 10 repeticiones y posteriormente se calcularon las métricas de número de puntos y SCC descritas anteriormente.

| Valor de los parámetros a ajustar | # puntos | SCC |
|---|---|---|
| (100, 300, 15) | 5.354 | 0.345232 |
| (100, 300, 20) | 5.218 | 0.329387 |
| (100, 500, 15) | 6.768 | 0.532412 |
| (100, 500, 20) | 6.421 | 0.518236 |
| (200, 300, 15) | 7.973 | 0.563467 |
| (200, 300, 20) | 7.215 | 0.533827 |
| (200, 500, 15) | 7.431 | 0.534423 |
| (200, 500, 20) | 6.917 | 0.482853 |

Tabla 5.3: Promedios de las métricas obtenidas para cada conjunto de parámetros.

En la tabla 5.3, es evidente que la mejor combinación de parámetros, tanto en el número de puntos no dominados encontrados así como en el hipervolumen cubierto, es la que tiene 200 elementos en la población inicial, tiene una probabilidad del 15 % de mutación y su criterio de paro son 300 iteraciones. Es esta la combinación de parámetros que se utilizará en toda la experimentación siguiente.

## 5.5 Análisis de la pertenencia de las componentes del algoritmo propuesto

En las secciones siguientes se pretende evaluar algunos componentes de nuestra implementación del algoritmo NSGA-II. Todas las implementaciones se ejecutaron en una PC bajo el sistema operativo Windows 8 con procesador AMD A6-4400 APU with Radeon(tm) HD Graphics 2.70 GHz y 6 GB de RAM. Los códigos fueron creados y procesados bajo el software libre Dev-C++.

Se encontrará inicialmente una evaluación de los dos procesos de cruzamiento anteriormente propuestos, así como la evaluación del proceso de mutación y el costo computacional que implica su uso.

Finalmente, se encontrará la comparación entre los frentes de Pareto exactos y los frentes aproximados que obtuvimos con nuestra implementación final de NSGA-II.

### 5.5.1 Cruzamiento 1 vs cruzamiento 2

Como se recordará en la sección 4.4 el capítulo 4 se proponen dos tipos de cruzamiento OX.

- Cruzamiento 1: OX clásico donde se obtienen dos hijos.
- Cruzamiento 2: OX modificado donde se obtienen ocho hijos.

En esta sección, se analizarán los frentes aproximados de Pareto obtenidos con uno y otro cruzamiento, para así seleccionar el que tenga un mejor desempeño. Para esta comparación entre los cruzamientos se utilizaron las instancias TRP-S reportadas por Salehipour et al [42].

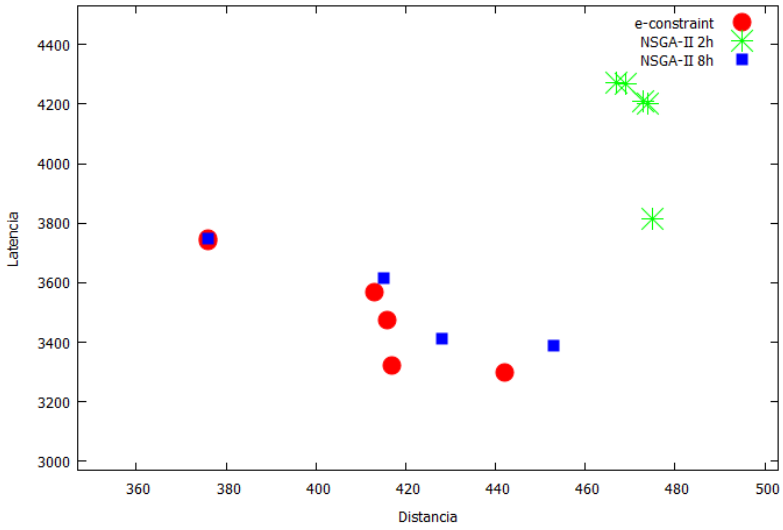



Figura 5.6: Comparación entre ambos procesos de cruzamiento para la instancia TRP-S20-14

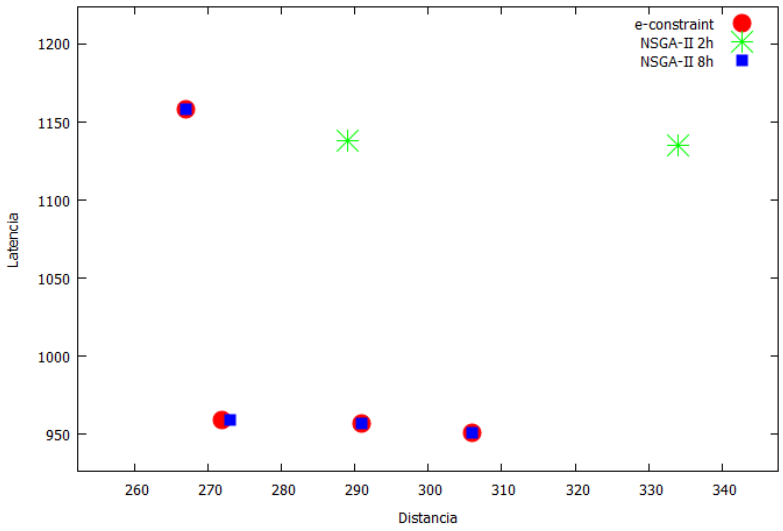



Figura 5.7: Comparación entre ambos procesos de cruzamiento para la instancia TRP-S10-20

En las imágenes 5.6 y 5.7, pueden observarse los frentes de pareto obtenidos por medio del $\epsilon$-constraint, así como los frentes aproximados obtenidos con el proceso NSGA-II con las dos variaciones de cruzamiento para la instancia TRP-S20-14 y

para la instancia TRP-S10-20 respectivamente. La cruza OX que genera dos hijos es nombrada como *NSGA-II2h* mientras que el cruzamiento que genera a los 8 hijos es nombrado como *NSGA-II8h*.

| Cruza | Instancia | n | CPU time |
|---|---|---|---|
| NSGA-II 2h | TRP-S | 10 | 34.73722222 |
| | | 20 | 49.0071875 |
| NSGA-II 8h | TRP-S | 10 | 142.0481765 |
| | | 20 | 456.8747059 |

Tabla 5.4: Comparación de los tiempos de cómputo para ambos procesos de cruzamiento.

Visualmente es evidente que el proceso NSGA-II 8H tiene un mejor desempeño en relación al proceso NSGA-II2H. En la tabla 5.4 se puede observar el tiempo computacional promedio que tarda cada cruza. El proceso NSGA-II 8H tarda más tiempo que el proceso NSGA-II 2H, sin embargo, brinda mejores resultados. Cabe destacar que todos los tiempos están en segundos.

En general, el proceso NSGA-II 8H tuvo un mejor desempeño. Esto es evidente en las tablas 5.5 y 5.6, en donde se reflejan los promedios de todas las instancias de 10 clientes así como en las tablas 5.7 y 5.8, en donde se reflejan los promedios de todas las instancias de 20 clientes. En estas tablas puede observarse que el proceso de cruza NSGA-II 2H se queda muy lejos del frente exacto, mientras que el proceso NSGA-II 8H se acerca más al frente exacto brindado por el $\epsilon$-constraint. También en la cobertura de los frentes queda claro que el frente obtenido por el proceso NSGA-II 8H domina al frente obtenido por el proceso NSGA-II 2H.

| | # puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| NSGA-II 2H | 2.1 | 0.34656665 | 0.862309125 | 0.334774353 |
| NSGA-II 8H | 3.5 | 0.785812 | 0.5625144 | 0.00513463 |

Tabla 5.5: Comparación entre ambos cruzamientos para instancias de 10 clientes.

| | NSGA-II 2H | NSGA-II 8H |
|---|---|---|
| NSGA-II 2H | 0 | 0 |
| NSGA-II 8H | 0.552083375 | 0 |

Tabla 5.6: Comparación entre las métricas C(A,B) y C(B,A) entre ambos cruzamientos para instancias de 10 clientes.

En general, podemos concluir que el proceso de cruza NSGA-II 8H es mejor que el proceso NSGA-II 2H. Por este motivo, utilizaremos esta cruza para la implementación final del NSGA-II y será la que se utilice en los experimentos siguientes.

### 5.5.2 Mutación vs No Mutación

En la sección 4.4 se explicó que la mutación que se propone consiste en una búsqueda local guiada por el objetivo de latencia. En el siguiente experimento pretendemos investigar la conveniencia de la inclusión de esta mutación en el algoritmo propuesto. Para esta comparación se tomaron nuevamente las instancias TRP-S reportadas por Salehipour et al [42].

En la imagen 5.8, se puede observar el frente de pareto exacto obtenido por el algoritmo $\epsilon$-constraint, así como los frentes brindados por la implementación del NSGA-II que incluye la cruza NSGA-II 8H y no incluye mutación (a esta implementación se le denominará igual que a la cruza, esto debido a que no incluye el proceso de mutación).

| | # Puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| NSGA-II 2H | 3.058823529 | 0.196620441 | 0.261487 | 0.658949 |
| NSGA-II 8H | 4.352941176 | 0.794842706 | 0.181304965 | 0.10107244 |

Tabla 5.7: Comparación entre ambos cruzamientos para instancias de 20 clientes.

| | NSGA-II 2H | NSGA-II 8H |
|---|---|---|
| NSGA-II 2H | 0 | 0.014705882 |
| NSGA-II 8H | 0.852941176 | 0 |

Tabla 5.8: Comparación entre las métricas C(A,B) y C(B,A) entre ambos cruzamientos para instancias de 20 clientes.

Así como la implementación que incluye la cruza NSGA-II 8H junto a la mutación (esta implementación es denominada NSGA-II 8H BL).

Visualmente, puede observarse que al utilizar la cruza NSGA-II 8H y el proceso de mutación juntos, se mejora considerablemente el frente obtenido.

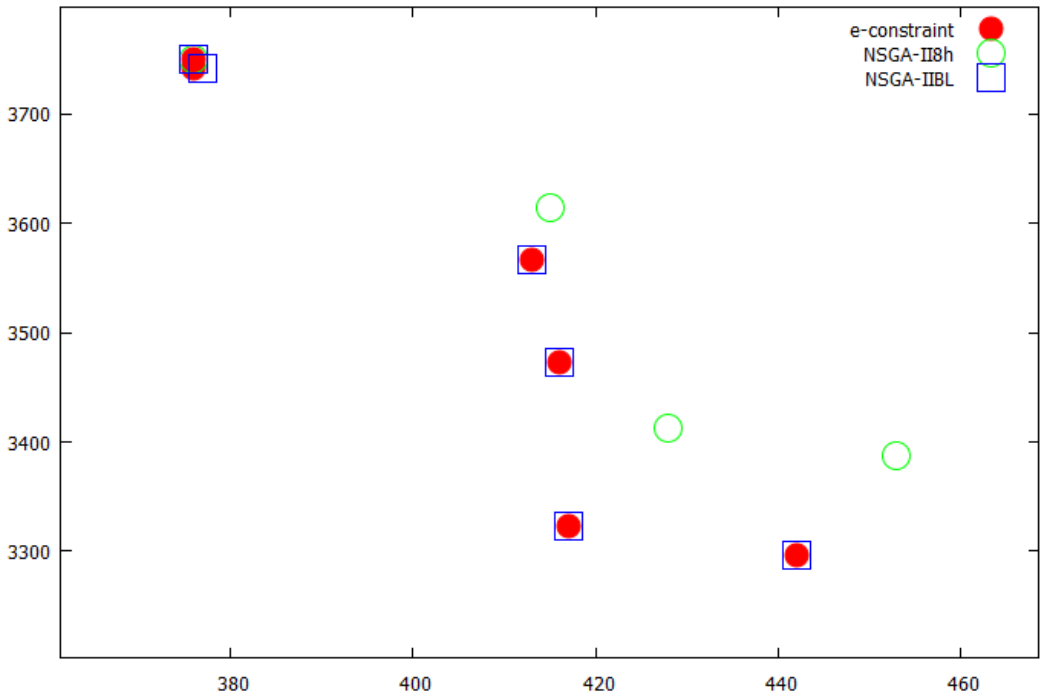



Figura 5.8: Comparación entre el proceso de cruzamiento NSGA-II8h con y sin mutación para la instancia TRP-S20-14

En general, al incluir el proceso de mutación propuesto se obtuvieron mejores resultados. En las tablas 5.9 y 5.10 se pueden observar los promedios obtenidos para las instancias de 10 clientes. En las tablas 5.11 y 5.12 pueden observarse los promedios para las instancias de tamaño 20. En general, es visible que el proceso con la búsqueda local logra mejorar el frente generado sin ella y podemos reafirmar que esta unión de procesos genera un mejor frente, capaz de acercarse muchísimo al frente exacto.

| | # Puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| NSGA-II 8H | 3.4375 | 0.771848188 | 0.862309125 | 0.334774353 |
| NSGA-II 8H BL | 3.5 | 0.785812 | 0.5625144 | 0.00513463 |

Tabla 5.9: Métricas de desempeño para las instancias de 10 clientes

| | NSGA-II 8H | NSGA-II 8H BL |
|---|---|---|
| NSGA-II 8H | 0 | 0 |
| NSGA-II 8H BL | 0.552083375 | 0 |

Tabla 5.10: Métricas de desempeño para las instancias de 10 clientes

| | # Puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| NSGA-II 8H | 4.294117647 | 0.794842706 | 0.181304965 | 0.10107244 |
| NSGA-II 8H BL | 6.294117647 | 0.866167059 | 0.171604418 | 0.048832126 |

Tabla 5.11: Métricas de desempeño para las instancias de 20 clientes

| | NSGA-II 8H | NSGA-II 8H BL |
|---|---|---|
| NSGA-II 8H | 0 | 0.030995476 |
| NSGA-II 8H BL | 0.374369765 | 0 |

Tabla 5.12: Métricas de desempeño para las instancias de 20 clientes

| Cruza | Instancia | n | CPU time |
|---|---|---|---|
| NSGA-II 8h | TRP-S | 10 | 49.0071875 |
| | | 20 | 456.8747059 |
| NSGA-II 8hB | TRP-S | 10 | 264.5543125 |
| | | 20 | 1286.348 |

Tabla 5.13: Comparación de los tiempos de cómputo para el proceso de cruzamiento NSGA-II8h sin mutación y con mutación.

En la tabla 5.13 se pueden observar los tiempos de cómputo que tardan ambas implementaciones. Como es visible, el proceso de cruza unido a la mutación

brinda mejores resultados a un costo computacional más elevado pero posiblemente aceptable. Cabe destacar que todos los tiempos están en segundos.

Debido al desempeño que tuvo el proceso NSGA-II 8H unido a la mutación, esta será la elección para nuestra implementación final del NSGA-II.

## 5.6 Evaluación del algoritmo propuesto

Teniendo en cuenta los resultados de los experimentos descritos en las secciones 5.4.1 y 5.4.2, el algoritmo definitivo contará con el cruzamiento OX modificado (cruzamiento 2) e incluirá la mutación. Para evaluar la calidad de los frentes aproximados que se obtienen, se comparará con los frentes exactos obtenidos para las instancias TRP-S así como con las instancias GTRP-$S_0$.

En esta parte de la experimentación se utilizaron 20 instancias de 10 clientes y 20 intancias de 20 clientes, dando un total de 40 instancias de Salehipour et al [42], así como 25 instancias de 10 clientes, 25 instancias de 15 clientes y 25 instancias de 20 clientes, dando un total de 75 instancias de Angel-Bello et al [1].

Los parámetros utilizados son los parámetros que fueron encontrados con el diseño factorial $2^3$ que se realizó con anterioridad, es decir, la población inicial se consideró con un tamaño de 200 soluciones. La probabilidad de mutación es del 15 % y el número de iteraciones que correrá el algoritmo es 300.

Para las instancias de 10 clientes, se obtuvieron los siguientes resultados.

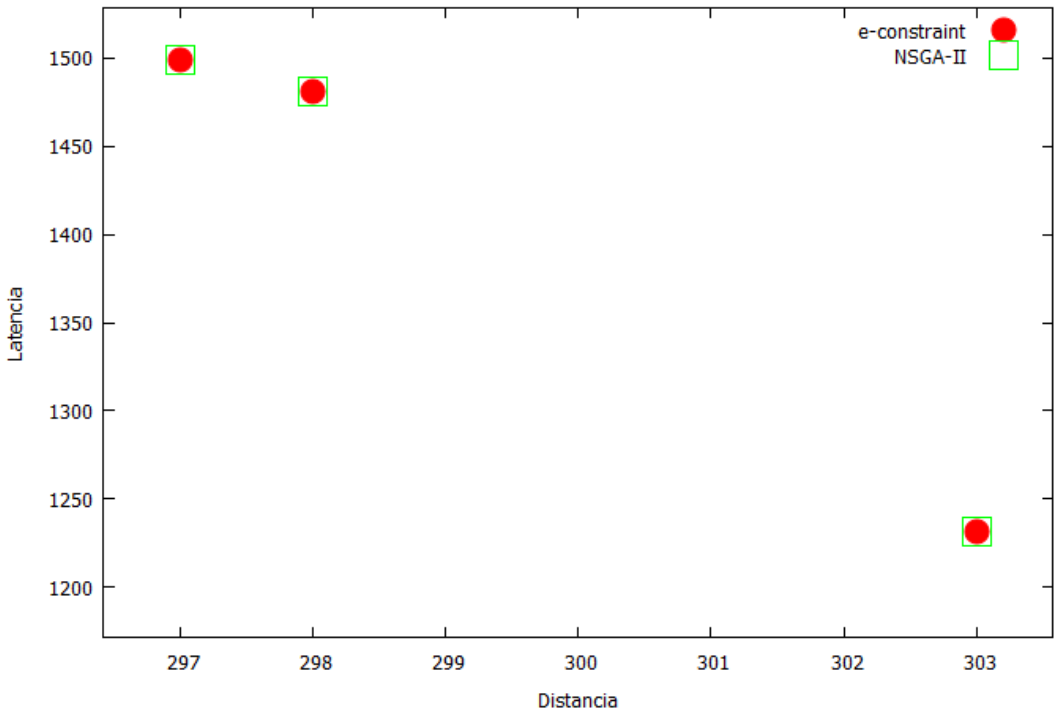



Figura 5.9: Frente de Pareto exacto y frente de Pareto aproximado para la instancia GTRP-$S_0$10-20

| | # Puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| Frente exacto | 3.208333333 | 0.620358917 | 0.76736325 | 0 |
| NSGA-II | 4.791666667 | 0.622906333 | 0.796515542 | 0.013583765 |

Tabla 5.14: Métricas de desempeño para el conjunto de instancias GTRP-$S_0$10

| | Frente exacto | NSGA-II |
|---|---|---|
| Frente exacto | 0 | 0.086652174 |
| NSGA-II | 0.084478261 | 0 |

Tabla 5.15: Métricas de desempeño para el conjunto de instancias GTRP-$S_0$10

En la figura 5.9, se puede observar gráficamente la comparación entre el

algoritmo exacto $\epsilon$-constraint y nuestra implementación del NSGA-II para la instancia GTRP-$S_0$10-20.

En general, en el conjunto de instancias GTRP-$S_0$10, nuestra implementación fue capaz de dominar cierto número de soluciones en el frente exacto (véase la tabla 5.15), esto es posible debido a que el algoritmo exacto solo puede encontrar las soluciones soportadas y nuestro algoritmo del NSGA-II fue capaz de encontrar algunas de las soluciones no soportadas que el método exacto no encontró. A pesar de que en promedio superamos por muy poco en el número de puntos no dominados en el frente, somos capaces de dar en promedio un mayor número de puntos no dominados en el frente (véase la tabla 5.14). A pesar de que el algoritmo exacto domina algunos de los puntos dados por el NSGA-II (véase la tabla 5.15), se logró un frente muy aproximado al exacto.

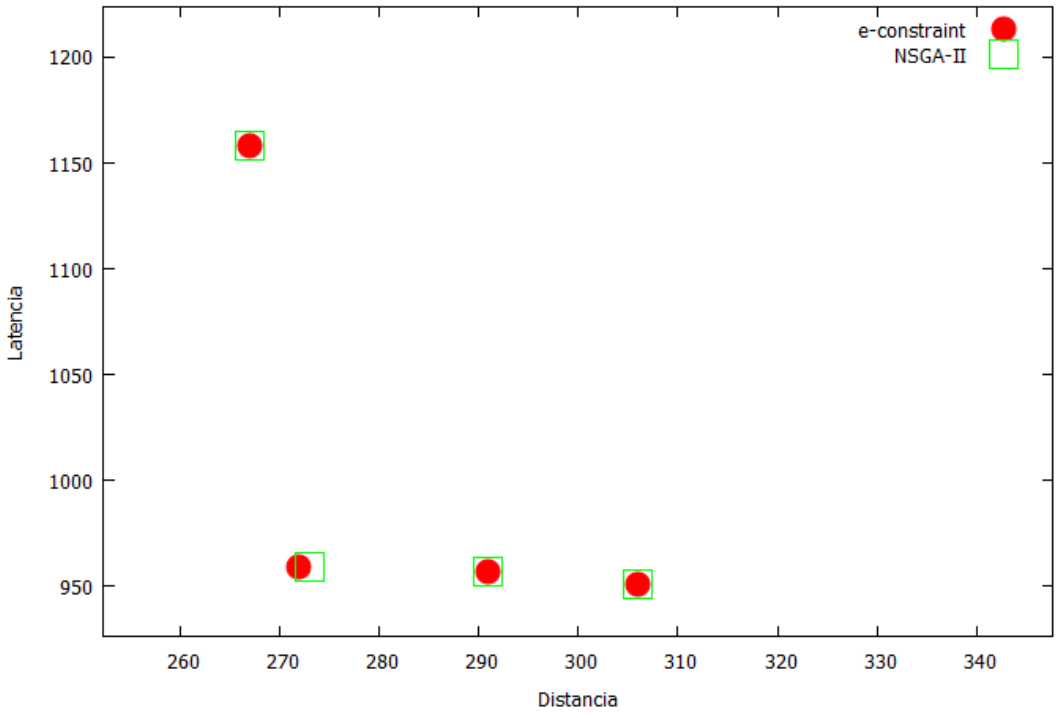



Figura 5.10: Frente de Pareto exacto y frente de Pareto aproximado para la instancia TRP-S10-20

| | # Puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| Frente exacto | 2.9 | 0.6198191 | 0.581653525 | 0 |
| NSGA-II | 4.176470588 | 0.771848188 | 0.862309125 | 0.034774353 |

Tabla 5.16: Métricas de desempeño para el conjunto de instancias TRP-S10

| | Frente exacto | NSGA-II |
|---|---|---|
| Frente exacto | 0 | 0.552083375 |
| NSGA-II | 0 | 0 |

Tabla 5.17: Métricas de desempeño para el conjunto de instancias TRP-S10

En la figura 5.10, se puede observar gráficamente la comparación entre el algoritmo exacto $\epsilon$-constraint y nuestra implementación del NSGA-II para la instancia TRP-S10-20.

En general, en el conjunto de instancias SalTRP10, nuestra implementación fue capaz de dar en promedio un mayor número de puntos no dominados en el frente (véase la tabla 5.16). También fue capaz de cubrir un mayor volumen de puntos dominados, y a pesar de que el algoritmo exacto domina algunos de los puntos dados por el NSGA-II (véase la tabla 5.17), se logró una buena aproximación al frente exacto.

Para las instancias de 15 clientes, se tuvieron los siguientes resultados.

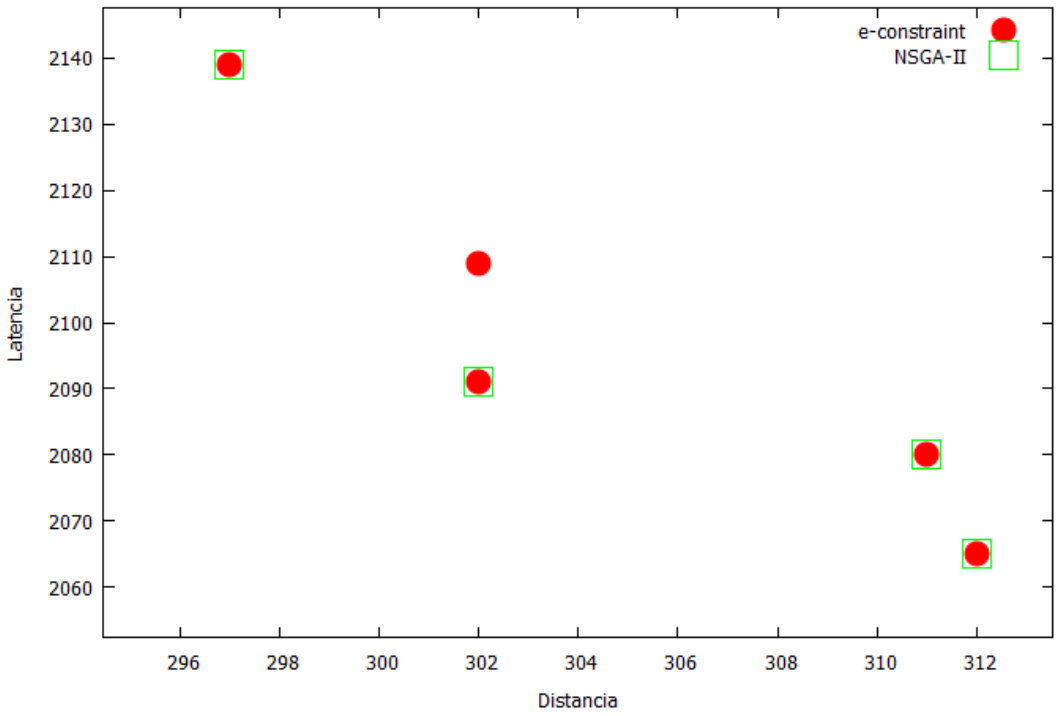



Figura 5.11: Frente de Pareto exacto y frente de Pareto aproximado para la instancia GTRP-$S_0$15-10

| | # Puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| Frente exacto | 4.782608696 | 0.630944522 | 0.548146435 | 0 |
| NSGA-II | 6.043478261 | 0.635335596 | 0.557950317 | 0.042534288 |

Tabla 5.18: Métricas de desempeño para el conjunto de instancias GTRP-$S_0$15

| | Frente exacto | NSGA-II |
|---|---|---|
| Frente exacto | 0 | 0.171304348 |
| NSGA-II | 0.079767833 | 0 |

Tabla 5.19: Métricas de desempeño para el conjunto de instancias GTRP-$S_0$15

En la figura 5.11 se puede observar la comparación entre el frente de Pareto exacto y el frente obtenido con nuestra implementación del NSGA-II en la instancia

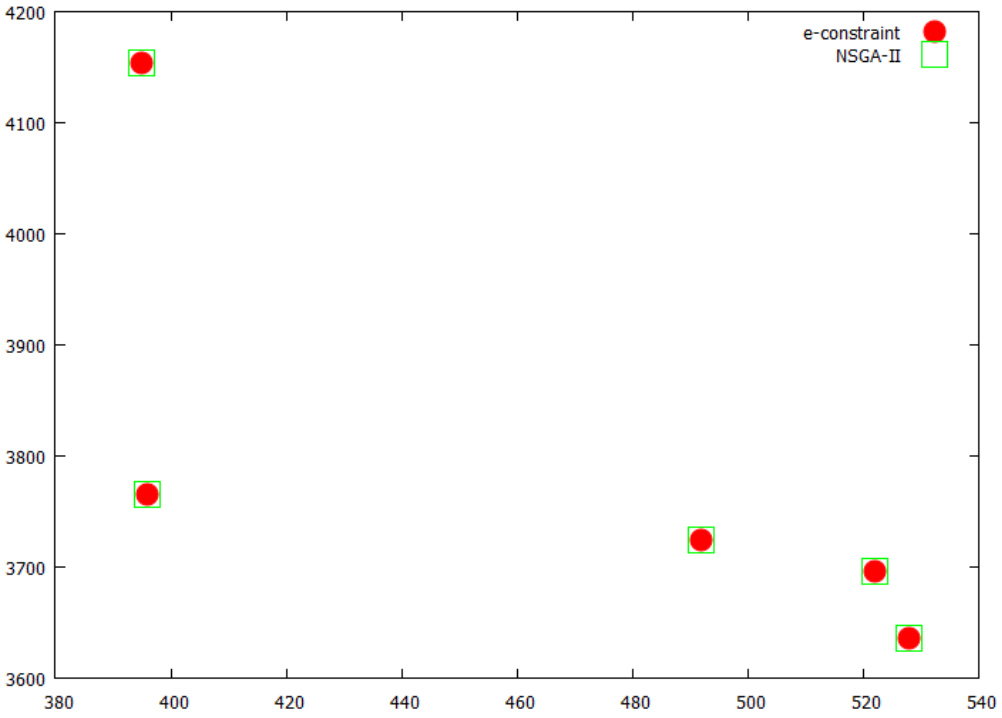



Figura 5.12: Frente de Pareto exacto y frente de Pareto aproximado para la instancia GTRP-$S_0$20-14

GTRP-$S_0$15. En general, nuestra implementación es capaz de superar en promedio el volumen de puntos dominados (véase a tabla 5.18). A pesar de que el algoritmo exacto domina algunos de los puntos dados por nuestra implementación del NSGA-II (véase la tabla 5.19), logramos obtener un frente muy aproximado al exacto.

Para las instancias de 20 clientes, se obtuvieron los resultados que se muestran a continuación. La figura 5.12 se puede observar la comparación entre el frente de pareto exacto y el frente obtenido con nuestro algoritmo NSGA-II en la instancia GTRP-$S_0$20-14.

En general, nuestra implementación es capaz de superar en promedio el volumen de puntos dominados (véase a tabla 5.20. El algoritmo exacto domina algunos de los puntos dados por el NSGA-II (véase la tabla 5.21), pero a pesar

| | # Puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| Frente exacto | 7.058235413 | 0.887456525 | 0.294112528 | 0 |
| NSGA-II | 8.647058824 | 0.889167069 | 0.160134184 | 0.021488326 |

Tabla 5.20: Métricas de desempeño para el conjunto de instancias GTRP-$S_0$20

| | Frente exacto | NSGA-II |
|---|---|---|
| Frente exacto | 0 | 0.18332691 |
| NSGA-II | 0 | 0 |

Tabla 5.21: Métricas de desempeño para el conjunto de instancias GTRP-$S_0$20

de eso se logró un frente muy aproximado al exacto.

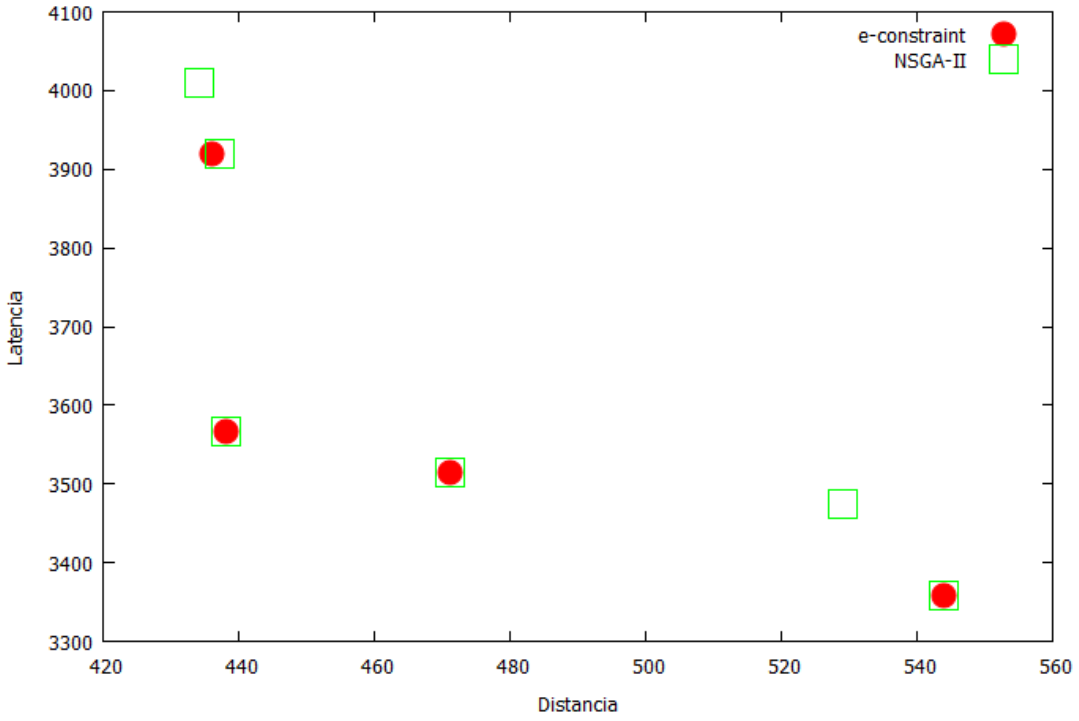



Figura 5.13: Frente de Pareto exacto y frente de Pareto aproximado para la instancia TRP-S20-10

| | # Puntos | SCC | k-Distancia | $M_1^*$ |
|---|---|---|---|---|
| Frente exacto | 5.470588235 | 0.858565765 | 0.225294118 | 0 |
| NSGA-II | 6.647058824 | 0.866167059 | 0.171604418 | 0.048832126 |

Tabla 5.22: Métricas de desempeño para el conjunto de instancias TRP-S20

| | Frente exacto | NSGA-II |
|---|---|---|
| Frente exacto | 0 | 0.197831706 |
| NSGA-II | 0.138401412 | 0 |

Tabla 5.23: Métricas de desempeño para el conjunto de instancias TRP-S20

En la figura 5.13 se puede observar la comparación entre el frente de pareto exacto y el frente obtenido con nuestra implementación del NSGA-II en la instancia TRP-S20-10. Se puede observar que en general nuestra implementación es capaz de superar en promedio el volumen de puntos dominados (véase a tabla 5.22). A pesar de que el algoritmo exacto domina algunos de los puntos dados por el NSGA-II (véase la tabla 5.23), se logró una buena aproximación al frente exacto.

En la tabla 5.24, se observan los tiempos computacionales que tarda nuestra implementación del NSGA-II. Se puede ver, que el tiempo que tarda es razonable. Todos nuestros tiempos están dados en segundos y promediados para cada conjunto de instancias.

| | Instancia | n | CPU time |
|---|---|---|---|
| NSGA-II | TRP-S | 10 | 264.5543125 |
| | | 20 | 1286.348 |
| NSGA-II | GTRP-$S_0$ | 10 | 294.3474 |
| | | 15 | 623.7521 |
| | | 20 | 1321.674 |

Tabla 5.24: Comparación de tiempos computacionales para las instancias de prueba.

Con estos experimentos se puede concluir que nuestra implementación del NSGA-II brinda en promedio, una buena aproximación al frente de pareto exacto en un tiempo de cómputo aceptable.

Capítulo 6

# Conclusiones

En este trabajo se ha estudiado un enfoque biobjetivo para el problema del reparador en el que se busca no sólo beneficiar a la empresa encargada de distribuir sus productos, sino que se busca también un beneficio para el cliente al minimizar su tiempo de espera para recibir el producto o el servicio que la empresa está encargada de suministrarle. En otras palabras, los objetivos que se buscó minimizar en este trabajo fueron tanto el objetivo de latencia como el objetivo de distancia.

Se propuso un modelo biobjetivo lineal entero. Este modelo fue utilizado para obtener los frentes de Pareto exactos utilizando el método de la $\epsilon$-restricción y se logró obtenerlos para instancias de hasta 20 clientes. Para instancias mayores, fue imposible obtener resultados de forma exacta debido a que el tiempo computacional se incrementa desmesuradamente.

Debido a la tardanza computacional para brindar los frentes exactos, se propuso un algoritmo basado en la metaheurística NSGA-II, el cual es un método genético elitista que busca aproximar los frentes de Pareto. Se propusieron dos métodos de cruza distintos, uno que generaba dos soluciones descendientes y otra que generaba ocho. Esta última resultó ser más eficiente, aún cuando tarda un mayor tiempo computacional.
También se investigó la conveniencia de incluir un proceso de mutación basado en una búsqueda local. Los experimentos mostraron que a pesar de que este aumenta

el tiempo de ejecución sí se mejora considerablemente la aproximación al frente de Pareto exacto.

La implementación final del algoritmo NSGA-II fue capaz de brindar, en promedio, frentes de Pareto con una buena aproximación a los frentes exactos. Podemos concluir entonces, que la metaheurística NSGA-II que proponemos es capaz de competir con los frentes de Pareto exactos brindados con el método exacto elegido.

## 6.1 Trabajo Futuro

Al estudiar un problema que no ha sido estudiado anteriormente, los puntos que pueden agregarse al estudio son muchos. Entre algunas cosas que podrían ser agregadas a este trabajo, se tienen las siguientes:

- Investigar los efectos que tendría en la aproximación del frente de Pareto que se obtiene, el uso de la búsqueda local guíada por la función objetivo de distancia, así como la interacción entre ambas búsquedas locales.
- Aplicar el estudio a una aplicación real.
- Investigar el uso de más de una ruta o más de un agente.

Apéndice A

# Frentes de Pareto Exactos

## A.1 Frentes de Pareto para las instancias

En este apéndice se reportan los frentes de pareto para las instancias reportadas en el trabajo de Salehipour et al [42] y para las instancias reportadas en el trabajo de Angel-Bello et al [1].

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| TRP-S10-01 | 1328 | 290 |
| | 1303 | 297 |
| TRP-S10-03 | 1237 | 308 |
| | 1233 | 309 |
| TRP-S10-05 | 1112 | 234 |
| | 1060 | 237 |
| | 978 | 267 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| TRP-S10-07 | 1178 | 326 |
| | 1163 | 327 |
| TRP-S10-08 | 1368 | 281 |
| | 1336 | 293 |
| | 1278 | 299 |
| | 1234 | 303 |
| TRP-S10-09 | 1440 | 307 |
| | 1420 | 312 |
| | 1402 | 324 |
| TRP-S10-10 | 1438 | 316 |
| | 1394 | 318 |
| | 1388 | 395 |
| TRP-S10-11 | 1449 | 294 |
| | 1441 | 308 |
| | 1433 | 319 |
| | 1427 | 330 |
| | 1405 | 343 |
| TRP-S10-12 | 1231 | 267 |
| | 1152 | 280 |
| | 1150 | 297 |
| TRP-S10-13 | 2113 | 368 |
| | 1567 | 368 |
| | 1561 | 404 |
| | 1531 | 408 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| TRP-S10-15 | 1260 | 284 |
| | 1250 | 287 |
| | 1193 | 288 |
| | 1160 | 298 |
| | 1150 | 301 |
| | 1148 | 303 |
| | 1129 | 311 |
| | 1087 | 313 |
| TRP-S10-16 | 1575 | 314 |
| | 1565 | 314 |
| | 1514 | 336 |
| | 1448 | 351 |
| | 1277 | 361 |
| | 1266 | 377 |
| | 1264 | 393 |
| TRP-S10-17 | 1219 | 270 |
| | 1058 | 271 |
| TRP-S10-18 | 1096 | 244 |
| | 1093 | 251 |
| | 1086 | 251 |
| | 1083 | 258 |
| TRP-S10-19 | 1416 | 338 |
| | 1394 | 366 |
| TRP-S10-20 | 1158 | 267 |
| | 959 | 272 |
| | 957 | 291 |
| | 951 | 306 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| TRP-S20-02 | 3460 | 373 |
| | 3337 | 386 |
| | 3286 | 415 |
| | 3248 | 426 |
| TRP-S20-04 | 3689 | 342 |
| | 3136 | 413 |
| | 3109 | 421 |
| | 2983 | 431 |
| TRP-S20-05 | 3348 | 336 |
| | 3322 | 338 |
| | 3310 | 355 |
| | 3248 | 397 |
| TRP-S20-06 | 4384 | 389 |
| | 3396 | 388 |
| | 3380 | 390 |
| | 3375 | 418 |
| | 3373 | 419 |
| | 3361 | 419 |
| | 3354 | 430 |
| | 3348 | 432 |
| | 3334 | 434 |
| | 3328 | 436 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| TRP-S20-07 | 3698 | 361 |
| | 3676 | 360 |
| | 3522 | 361 |
| | 3544 | 361 |
| | 3522 | 361 |
| | 3117 | 365 |
| | 3095 | 365 |
| | 3063 | 373 |
| | 3041 | 373 |
| | 3019 | 373 |
| | 3015 | 375 |
| | 2999 | 380 |
| | 2991 | 389 |
| | 2931 | 397 |
| | 2925 | 404 |
| | 2909 | 411 |
| | 2874 | 418 |
| | 2871 | 431 |
| | 2866 | 446 |
| | 2809 | 459 |
| TRP-S20-08 | 3898 | 375 |
| | 3602 | 375 |
| | 3601 | 412 |
| | 3510 | 422 |
| | 3488 | 424 |
| | 3461 | 436 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| TRP-S20-09 | 3738 | 399 |
| | 3723 | 405 |
| | 3657 | 409 |
| | 3624 | 413 |
| | 3617 | 474 |
| | 3608 | 478 |
| | 3580 | 483 |
| | 3475 | 511 |
| TRP-S20-10 | 3919 | 436 |
| | 3566 | 438 |
| | 3514 | 471 |
| | 3359 | 544 |
| TRP-S20-11 | 2966 | 341 |
| | 2948 | 340 |
| | 2945 | 341 |
| | 2931 | 342 |
| | 2930 | 342 |
| | 2916 | 343 |
| TRP-S20-13 | 3924 | 389 |
| | 3746 | 388 |
| | 3704 | 391 |
| | 3602 | 392 |
| | 3582 | 393 |
| | 3412 | 409 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| TRP-S20-14 | 3750 | 376 |
| | 3741 | 376 |
| | 3567 | 413 |
| | 3473 | 416 |
| | 3323 | 417 |
| | 3297 | 442 |
| TRP-S20-15 | 3057 | 336 |
| | 2862 | 338 |
| TRP-S20-16 | 3694 | 398 |
| | 3543 | 413 |
| | 3526 | 436 |
| | 3433 | 442 |
| TRP-S20-17 | 2988 | 337 |
| | 2926 | 362 |
| | 2925 | 418 |
| | 2924 | 419 |
| | 2913 | 420 |
| TRP-S20-18 | 3492 | 396 |
| | 3478 | 396 |
| | 3442 | 397 |
| | 3430 | 398 |
| | 3294 | 402 |
| | 3292 | 402 |
| | 3276 | 403 |
| | 3264 | 405 |
| | 3254 | 415 |
| | 3124 | 420 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| TRP-S20-19 | 3898 | 379 |
| | 3682 | 378 |
| | 3498 | 421 |
| | 3497 | 422 |
| | 3311 | 428 |
| | 3309 | 429 |
| | 3299 | 450 |
| TRP-S20-20 | 2906 | 382 |
| | 2902 | 383 |
| | 2798 | 396 |
| | 2796 | 398 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP10s0-01 | 1035 | 269 |
| | 965 | 280 |
| | 957 | 286 |
| GTRP10s0-02 | 1853 | 330 |
| | 1447 | 330 |
| GTRP10s0-03 | 1824 | 292 |
| | 1095 | 292 |
| GTRP10s0-05 | 1858 | 322 |
| | 1372 | 323 |
| | 1252 | 326 |
| GTRP10s0-06 | 1516 | 265 |
| | 1134 | 265 |
| | 1066 | 278 |
| | 954 | 287 |
| GTRP10s0-07 | 1014 | 222 |
| | 985 | 231 |
| GTRP10s0-08 | 1682 | 338 |
| | 1674 | 342 |
| | 1666 | 351 |
| | 1600 | 359 |
| | 1590 | 388 |
| | 1552 | 391 |
| | 1530 | 393 |
| GTRP10s0-09 | 1729 | 304 |
| | 1321 | 305 |
| | 1057 | 311 |
| | 1055 | 313 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP10s0-10 | 1503 | 308 |
| | 1455 | 326 |
| GTRP10s0-11 | 1811 | 350 |
| | 1689 | 350 |
| | 1683 | 352 |
| | 1459 | 358 |
| | 1446 | 398 |
| GTRP10s0-13 | 1452 | 317 |
| | 1401 | 317 |
| | 1393 | 327 |
| | 1375 | 341 |
| | 1367 | 352 |
| GTRP10s0-14 | 1705 | 279 |
| | 1285 | 298 |
| | 1277 | 322 |
| GTRP10s0-15 | 1433 | 324 |
| | 1411 | 442 |
| | 1410 | 443 |
| GTRP10s0-16 | 1111 | 237 |
| | 1108 | 238 |
| | 1101 | 254 |
| | 1077 | 258 |
| GTRP10s0-17 | 1321 | 246 |
| | 1051 | 246 |
| GTRP10s0-18 | 1492 | 294 |
| | 1448 | 294 |
| | 1340 | 326 |
| | 1328 | 332 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP10s0-19 | 1066 | 267 |
| | 1063 | 274 |
| GTRP10s0-20 | 1499 | 297 |
| | 1481 | 298 |
| | 1231 | 303 |
| GTRP10s0-21 | 1860 | 304 |
| | 1180 | 304 |
| | 1120 | 305 |
| | 1119 | 306 |
| GTRP10s0-22 | 1757 | 311 |
| | 1353 | 311 |
| | 1288 | 348 |
| GTRP10s0-23 | 1613 | 318 |
| | 1564 | 322 |
| | 1392 | 332 |
| | 1380 | 333 |
| | 1296 | 358 |
| GTRP10s0-24 | 1572 | 294 |
| | 1368 | 294 |
| GTRP10s0-25 | 1979 | 308 |
| | 1101 | 308 |
| | 1037 | 330 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP15s0-01 | 3051 | 373 |
| | 2544 | 372 |
| | 2451 | 376 |
| | 2439 | 378 |
| | 2302 | 430 |
| | 2256 | 438 |
| GTRP15s0-02 | 2691 | 333 |
| | 2173 | 346 |
| | 2056 | 356 |
| GTRP15s0-03 | 2792 | 392 |
| | 2786 | 397 |
| | 2588 | 417 |
| | 2582 | 422 |
| | 2540 | 430 |
| | 2537 | 436 |
| GTRP15s0-04 | 2315 | 341 |
| | 2232 | 355 |
| | 2207 | 397 |
| GTRP15s0-05 | 2293 | 343 |
| | 2266 | 405 |
| | 2217 | 411 |
| GTRP15s0-06 | 2379 | 324 |
| | 2261 | 335 |
| | 2256 | 336 |
| | 2254 | 336 |
| | 2232 | 347 |
| | 2207 | 360 |
| | 2172 | 367 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP15s0-07 | 3130 | 394 |
| | 2780 | 394 |
| | 2570 | 404 |
| | 2552 | 426 |
| | 2537 | 465 |
| | 2518 | 477 |
| | 2509 | 515 |
| GTRP15s0-08 | 2846 | 350 |
| | 2419 | 351 |
| | 2409 | 354 |
| | 2395 | 358 |
| | 2389 | 359 |
| | 2379 | 362 |
| | 2347 | 363 |
| | 2337 | 367 |
| | 2323 | 370 |
| | 2267 | 392 |
| GTRP15s0-09 | 2838 | 364 |
| | 2256 | 364 |
| | 2232 | 380 |
| | 1998 | 388 |
| GTRP15s0-10 | 2139 | 297 |
| | 2109 | 302 |
| | 2091 | 302 |
| | 2080 | 311 |
| | 2065 | 312 |
| GTRP15s0-11 | 2746 | 342 |
| | 2384 | 342 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP15s0-12 | 3112 | 369 |
| | 2437 | 370 |
| GTRP15s0-13 | 2462 | 342 |
| | 2414 | 357 |
| | 2156 | 359 |
| | 2034 | 366 |
| GTRP15s0-14 | 2714 | 377 |
| | 2710 | 406 |
| | 2698 | 410 |
| | 2684 | 418 |
| GTRP15s0-16 | 2971 | 396 |
| | 2969 | 396 |
| | 2933 | 396 |
| | 2543 | 403 |
| | 2524 | 418 |
| | 2457 | 421 |
| | 2373 | 445 |
| GTRP15s0-17 | 2516 | 323 |
| | 2344 | 324 |
| | 2320 | 328 |
| | 2292 | 329 |
| | 2278 | 333 |
| | 2226 | 338 |
| | 2205 | 369 |
| | 2133 | 387 |
| GTRP15s0-18 | 3180 | 362 |
| | 2536 | 362 |
| | 2250 | 361 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP15s0-19 | 2674 | 317 |
| | 2081 | 317 |
| | 2059 | 318 |
| | 1936 | 324 |
| | 1912 | 325 |
| GTRP15s0-20 | 2448 | 341 |
| | 2333 | 344 |
| | 2327 | 345 |
| | 2126 | 364 |
| | 2071 | 378 |
| GTRP15s0-21 | 3154 | 366 |
| | 2336 | 366 |
| | 2322 | 370 |
| | 2312 | 482 |
| | 2286 | 492 |
| GTRP15s0-22 | 2883 | 381 |
| | 2774 | 381 |
| | 2753 | 408 |
| | 2709 | 411 |
| | 2693 | 446 |
| | 2683 | 450 |
| | 2682 | 451 |
| GTRP15s0-23 | 2520 | 345 |
| | 2514 | 399 |
| | 2479 | 418 |
| GTRP15s0-24 | 2298 | 321 |
| | 2216 | 332 |
| GTRP15s0-25 | 2768 | 354 |
| | 2557 | 355 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP20s0-01 | 366 | 3921 |
| | 366 | 3399 |
| | 380 | 3250 |
| | 387 | 3135 |
| | 394 | 2999 |
| | 398 | 2995 |
| GTRP20s0-02 | 327 | 2715 |
| | 330 | 2212 |
| GTRP20s0-03 | 357 | 3571 |
| | 357 | 3569 |
| | 359 | 3550 |
| | 374 | 3506 |
| | 429 | 3409 |
| | 431 | 3395 |
| | 432 | 3394 |
| GTRP20s0-04 | 388 | 4360 |
| | 388 | 3420 |
| | 390 | 3210 |
| | 393 | 3157 |
| | 396 | 2943 |
| | 403 | 2929 |
| | 417 | 2911 |
| GTRP20s0-05 | 407 | 4188 |
| | 407 | 3972 |
| | 409 | 3948 |
| | 481 | 3816 |
| | 485 | 3789 |
| | 491 | 3735 |
| | 520 | 3648 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP20s0-06 | 387 | 3460 |
| | 431 | 3423 |
| GTRP20s0-07 | 405 | 4685 |
| | 405 | 3415 |
| | 409 | 3326 |
| | 412 | 3278 |
| | 423 | 3264 |
| | 436 | 3234 |
| | 444 | 3188 |
| GTRP20s0-08 | 383 | 4391 |
| | 382 | 3267 |
| | 385 | 3223 |
| | 447 | 3206 |
| | 478 | 3192 |
| | 487 | 3162 |
| | 495 | 3130 |
| | 501 | 3093 |
| GTRP20s0-09 | 417 | 4871 |
| | 417 | 3469 |
| | 449 | 3417 |
| | 451 | 3399 |
| | 465 | 3394 |
| GTRP20s0-10 | 425 | 4377 |
| | 425 | 4113 |
| | 435 | 4059 |
| | 443 | 4037 |
| | 443 | 4029 |
| | 551 | 4005 |
| | 552 | 3969 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP20s0-11 | 380 | 3353 |
| | 448 | 3327 |
| | 451 | 3231 |
| | 455 | 3210 |
| | 454 | 3197 |
| | 456 | 3023 |
| | 459 | 2987 |
| GTRP20s0-12 | 422 | 3725 |
| | 427 | 3707 |
| | 511 | 3684 |
| | 517 | 3672 |
| GTRP20s0-13 | 450 | 4828 |
| | 449 | 4172 |
| | 456 | 4144 |
| | 462 | 4122 |
| | 502 | 4107 |
| | 506 | 4074 |
| | 508 | 4003 |
| GTRP20s0-14 | 395 | 4154 |
| | 396 | 3766 |
| | 492 | 3725 |
| | 522 | 3696 |
| | 528 | 3636 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP20s0-15 | 370 | 3692 |
| | 372 | 3675 |
| | 380 | 3327 |
| | 391 | 3284 |
| | 393 | 3251 |
| | 411 | 3212 |
| | 428 | 3206 |
| | 428 | 3192 |
| | 442 | 3183 |
| GTRP20s0-16 | 328 | 3555 |
| | 328 | 3025 |
| | 352 | 3003 |
| GTRP20s0-17 | 350 | 3966 |
| | 350 | 3034 |
| | 352 | 2833 |
| | 389 | 2750 |
| | 410 | 2743 |
| GTRP20s0-18 | 404 | 3656 |
| | 403 | 3652 |
| | 405 | 3563 |
| | 412 | 3516 |
| | 431 | 3483 |
| | 431 | 3479 |
| | 439 | 3329 |
| GTRP20s0-19 | 365 | 3954 |
| | 365 | 3346 |
| | 366 | 3322 |
| | 377 | 3312 |
| | 384 | 3304 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP20s0-20 | 342 | 3297 |
| | 343 | 3280 |
| | 343 | 3273 |
| | 353 | 3270 |
| | 357 | 3231 |
| | 358 | 3247 |
| | 359 | 3230 |
| | 358 | 3214 |
| | 430 | 3186 |
| | 431 | 3172 |
| GTRP20s0-21 | 350 | 2853 |
| | 352 | 2850 |
| | 363 | 2823 |
| | 372 | 2809 |
| | 382 | 2805 |
| | 404 | 2761 |
| GTRP20s0-22 | 368 | 3948 |
| | 367 | 3412 |
| | 377 | 3409 |
| | 404 | 3149 |
| GTRP20s0-23 | 364 | 4125 |
| | 364 | 3155 |
| | 370 | 2911 |
| | 406 | 2838 |
| | 430 | 2809 |
| | 472 | 2799 |
| | 475 | 2775 |

| Nombre | Obj $F_1$ | Obj $F_2$ |
|---|---|---|
| GTRP20s0-24 | 375 | 3567 |
| | 374 | 3563 |
| | 376 | 3561 |
| | 382 | 3499 |
| | 390 | 3413 |
| | 439 | 3384 |
| GTRP20s0-25 | 374 | 4183 |
| | 375 | 3317 |
| | 399 | 3314 |
| | 404 | 3273 |
| | 414 | 3271 |
| | 420 | 3224 |
| | 429 | 3208 |

# Bibliografía


[1] Angel-Bello, F., A. Alvarez y I. Garcia, «Two improved formulations for the minimum latency problem», *Applied Mathematical Modelling*, **37**(4), págs. 2257 – 2266, 2013, URL `http://www.sciencedirect.com/science/article/pii/S0307904X12003459`.

[2] Angel-Bello, F., I. Martinez-Salazar y A. Alvarez, «Minimizing waiting times in a route design problem with multiple use of a single vehicle», en *in INOC, 2003*, 2013.

[3] Applegate, D. L., R. E. Bixby, V. Chvatal y W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*, Princeton University Press, 2007.

[4] Archer, A. y A. Blasiak, «Improved approximation algorithms for the minimum latency problem via prize-collecting strolls», en *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, págs. 429–447, 2010.

[5] Archer, A., A. Levin y D. P. Williamson, «A Faster, Better Approximation Algorithm for the Minimum Latency Problem», *Informe técnico*, 2003.

[6] BANDYOPADHYAY, S., «Modified NSGA-II for a Bi-Objective Job Sequencing Problem», *Intelligent Information Management*, **4**(6), págs. 319–329, 2012.

[7] BENOIT, A., V. REHN-SONIGO y Y. ROBERT, «Optimizing Latency and Reliability of Pipeline Workflow Applications», , 2008.

[8] BLUM, A., P. CHALASANI, D. COPPERSMITH, B. PULLEYBLANK, P. RAGHAVAN y M. SUDAN, «The minimum latency problem», en *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, ACM, New York, NY, USA, págs. 163–171, 1994, URL `http://doi.acm.org/10.1145/195058.195125`.

[9] BÉRUBÉ, J.-F., M. GENDREAU y J.-Y. POTVIN, «An exact $\epsilon$-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits», *European Journal of Operational Research*, **194**(1), págs. 39 – 50, 2009, URL `http://www.sciencedirect.com/science/article/pii/S0377221707012106`.

[10] CARPANETO, G. y P. TOTH, «Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem», *Management Science*, **26**(7), págs. 736–743, 1980.

[11] CHAKRABARTY, D. y C. SWAMY, «Facility Location with Client Latencies: Linear Programming Based Techniques for Minimum Latency Problems.», en O. Gunluk y G. J. Woeginger (editores), *IPCO*, *Lecture Notes in Computer Science*, tomo 6655, Springer, págs. 92–103, 2011.

[12] CHEKURI, C., R. MOTWANI, B. NATARAJAN y C. STEIN, «Approximation Techniques for Average Completion Time Scheduling», en *SODA*, págs. 609–618, 1997.

[13] CIRASELLA, J., D. S. JOHNSON, L. A. MCGEOCH y W. ZHANG, «The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests», , 2000.

[14] COZZOLINO, A., *Humanitarian Logistics: Cross-Sector Cooperation in Disaster Relief Management*, SpringerBriefs in Business, Springer, 2012, URL `http://books.google.com.mx/books?id=rC2zHgjzCzQC`.

[15] CROES, G. A., «A Method for Solving Traveling-Salesman Problems», *Operations Research*, **6**(6), págs. 791–812, 1958.

[16] DAVID S. JOHNSON, L. A. M., *The Traveling Salesman Problem: A Case Study in Local Optimization*, John Wiley and Sons, London, 1997.

[17] DEB, K., A. PRATAP, S. AGARWAL y T. MEYARIVAN, «A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II», *IEEE Transactions on Evolutionary Computation*, **6**, págs. 182–197, 2000.

[18] DORIGO, M. y L. M. GAMBARDELLA, «Ant Colonies for the Traveling Salesman Problem», , 1997.

[19] FRIGGSTAD, Z., M. R. SALAVATIPOUR y Z. SVITKINA, «Asymmetric traveling salesman path and directed latency problems», en *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, págs. 419–428, 2010, URL `http://dl.acm.org/citation.cfm?id=1873601.1873636`.

[20] GOUVEIA, L. y S. VO[SS], «A classification of formulations for the (time-dependent) traveling salesman problem», *European Journal of Operational Research*, **83**(1), págs. 69–82, May 1995, URL `http://ideas.repec.org/a/eee/ejores/v83y1995i1p69-82.html`.

[21] GOYAL, S., «A Survey on Travelling Salesman Problem», en *Proc. 43rd Midwest Instruction and Computing Symposium (MICS 2010)*, 2010.

[22] GRÖTSCHEL, M. y O. HOLLAND, «Solution of large-scale symmetric travelling salesman problems», *Math. Program.*, **51**(2), págs. 141–202, julio 1991, URL http://dx.doi.org/10.1007/BF01586932.

[23] GUPTA, D., S. SHARMA y S. AGGARWAL, «Bi-objective scheduling on parallel machines with uncertain processing time», **3**(2), 2012.

[24] HUANG, S. C.-H., H. DU y E.-K. PARK, «Minimum-latency Gossiping in Multi-hop Wireless Networks», en *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '08, ACM, New York, NY, USA, págs. 323–330, 2008.

[25] HUANG, S. C.-H., H. DU y E.-K. PARK, «Minimum-latency gossiping in multi-hop wireless networks», en *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, ACM, New York, NY, USA, págs. 323–330, 2008, URL http://doi.acm.org/10.1145/1374618.1374662.

[26] JOZEFOWIEZ, N., F. GLOVER y M. LAGUNA, «Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits», *Journal of Mathematical Modelling and Algorithms*, **7**(2), págs. 177–195, 2008, URL http://dx.doi.org/10.1007/s10852-008-9080-2.

[27] JOZEFOWIEZ, N., F. GLOVER y M. LAGUNA, «Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits.», *J. Math. Model. Algorithms*, **7**(2), págs. 177–195, 2008, URL http://dblp.uni-trier.de/db/journals/jmma/jmma7.html#JozefowiezGL08.

[28] LAPORTE, G., «The traveling salesman problem: An overview of exact and approximate algorithms», *European Journal of Operational Research*, **59**(2),

págs. 231–247, June 1992, URL http://ideas.repec.org/a/eee/ejores/v59y1992i2p231-247.html.

[29] LAPORTE, G., «A Short History of the Traveling Salesman Problem», , 2006.

[30] LENSTRA, J. K. y A. H. G. R. KAN, «Some Simple Applications of the Travelling Salesman Problem», *Operational Research Quarterly (1970-1977)*, **26**(4), págs. 717–733, 1975, URL http://dx.doi.org/10.2307/3008306.

[31] LI, H., Q. S. HUA, C. WU y F. C. M. LAU, «Minimum-latency aggregation scheduling in wireless sensor networks under physical interference model», en *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, MSWIM '10, ACM, New York, NY, USA, págs. 360–367, 2010, URL http://doi.acm.org/10.1145/1868521.1868581.

[32] LI, H., Q. S. HUA, C. WU y F. C. M. LAU, «Minimum-latency aggregation scheduling in wireless sensor networks under physical interference model», en *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, MSWIM '10, ACM, New York, NY, USA, págs. 360–367, 2010, URL http://doi.acm.org/10.1145/1868521.1868581.

[33] MÉNDEZ-DÍAZ, I., P. ZABALA y A. LUCENA, «A new formulation for the Traveling Deliveryman Problem», *Discrete Appl. Math.*, **156**(17), págs. 3223–3237, octubre 2008, URL http://dx.doi.org/10.1016/j.dam.2008.05.009.

[34] NGUEVEU, S. U., C. PRINS y R. WOLFLER CALVO, «An Effective Memetic Algorithm for the Cumulative Capacitated Vehicle Routing Problem», *Comput. Oper. Res.*, **37**(11), págs. 1877–1885, noviembre 2010, URL http://dx.doi.org/10.1016/j.cor.2009.06.014.

[35] OLIVER, I. M., D. J. SMITH y J. R. C. HOLLAND, «A Study of Permutation Crossover Operators on the Traveling Salesman Problem», en *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, págs. 224–230, 1987, URL http://dl.acm.org/citation.cfm?id=42512.42542.

[36] ORMAN, A. J. y H. P. WILLIAMS, «A survey of different integer programming formulations of the travelling salesman problem», *Operational Research working papers,LSEOR*, **4**(67), págs. 736–743, 2004.

[37] PICARD, J. C. y M. QUEYRANNE, «The Time-dependent Traveling Salesman Problem and its Application to the Tardiness Problem in One-machine Scheduling», *Operations Research*, **26**(1), págs. 86–110, 1978.

[38] PLANTE, R. D., T. J. LOWE y R. CHANDRASEHARAN, «The product matrix traveling salesman problem: an application and solution heuristic», *Oper. Res.*, **35**(5), págs. 772–783, septiembre 1987, URL http://dx.doi.org/10.1287/opre.35.5.772.

[39] QADIR, J., C. T. CHOU, A. MISRA y J. G. LIM, «Minimum Latency Broadcasting in Multiradio, Multichannel, Multirate Wireless Meshes», *IEEE Transactions on Mobile Computing*, **8**(11), págs. 1510–1523, noviembre 2009, URL http://dx.doi.org/10.1109/TMC.2009.68.

[40] RIERA-LEDESMA, J. y J. J. SALAZAR-GONZÁLEZ, «The biobjective travelling purchaser problem», *European Journal of Operational Research*, **160**(3), págs. 599 – 613, decision Analysis and Artificial Intelligence, 2005, URL http://www.sciencedirect.com/science/article/pii/S037722170300660X.

[41] ROMERO, C., *Teoría de la decisión multicriterio: conceptos, técnicas y aplicaciones*, Alianza Universidad textos, Alianza Editorial, 1993.

[42] SALEHIPOUR, A., K. SORENSEN, P. GOOS y O. BRAYSY, «Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem», *4OR*, **9**, págs. 189–209, 2011, URL http://dx.doi.org/10.1007/s10288-011-0153-0.

[43] SARUBBI, J., H. LUNA y G. MIRANDA, «Minimum latency problem as a shortest path problem with side constraints», , 2008.

[44] SASTRY, K., D. GOLDBERG y G. KENDALL, «Genetic Algorithms», , 2005.

[45] SILVA, M. M., A. SUBRAMANIAN, T. VIDAL y L. S. OCHI, «A simple and effective metaheuristic for the Minimum Latency Problem», *European Journal of Operational Research*, págs. 513–520, 2012.

[46] TAVAKKOLI-MOGHADDAM, R., A. RAHIMI-VAHED y A. H. MIRZAEI, «A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: Weighted mean completion time and weighted mean tardiness», *Information Sciences*, **177**(22), págs. 5072 – 5090, 2007, URL http://www.sciencedirect.com/science/article/pii/S0020025507002861.

[47] THOMAS, L., A. AMD KOPCZAK, «From logistics to supply chain management: The path forward in the humanitarian sector», , 2005.

[48] WU, B. Y., Z.-N. HUANG y F.-J. ZHAN, «Exact algorithms for the minimum latency problem», *Inf. Process. Lett.*, **92**(6), págs. 303–309, diciembre 2004, URL http://dx.doi.org/10.1016/j.ipl.2004.09.009.

[49] ZITZLER, E., *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, Tesis Doctoral, ETH Zurich, Switzerland, 1999.

[50] ZITZLER, E., M. LAUMANNS y L. THIELE, «SPEA2: Improving the Strength Pareto Evolutionary Algorithm», *Informe técnico*, 2001.

[51] Zitzler, E. y L. Thiele, «Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach», , 1999.

# Ficha autobiográfica

Nancy Aracely Arellano Arriaga

Candidata para el grado de Maestría en Ciencias en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Un enfoque biobjetivo al problema del reparador.

Nací un 26 de septiembre de 1989 en el municipio de Piedras Negras, Coahuila. Viví alrededor de dos semanas ahí y cerca de 22 años en Saltillo, Coahuila así que también me considero saltillense. Soy orgullosamente Ateneísta de la generación 2004-2006, y en el 2010 me gradué como Licenciada en Matemáticas Aplicadas en la Universidad Autónoma de Coahuila. Actualmente, soy estudiante del Posgrado en Ingeniería de Sistemas de la Universidad Autónoma de Nuevo León en la Facultad de Ingeniería Mecánica y Eléctrica en donde realicé este trabajo de tesis bajo la supervisión y enseñanza de la Dra. Ada Margarita Álvarez Socarrás.