METODOLOGIA DE SOLUCIÓN HÍBRIDA: HEURISTICA-METAHEURISTICA-ENUMERACIÓN IMPLICITA 1-0 PARA EL PROBLEMA DE RUTEAMIENTO DE VEHICULOS CAPACITADO (CVRP)

David Escobar Vargas

1088304663

Universidad Tecnológica de Pereira

Facultad de Ingeniería

Programa de Ingeniería de Sistemas y Computación

Noviembre 2014

METODOLOGIA DE SOLUCIÓN HÍBRIDA: HEURISTICA-METAHEURISTICA-ENUMERACIÓN IMPLICITA 1-0 PARA EL PROBLEMA DE RUTEAMIENTO DE VEHICULOS CAPACITADO (CVRP)

David Escobar Vargas

1088304663

TRABAJO DE GRADO

Requisito parcial para optar al título de Ingeniero de Sistemas y Computación

Director

MAURICIO GRANADA ECHEVERRI

Ph.D en Ingeniería Eléctrica

Universidad Tecnológica de Pereira

Facultad de Ingeniería

Programa de Ingeniería de Sistemas y Computación

Noviembre 2014

A Laura, mis padres y mis hermanos

AGRADECIMIENTOS

- A la Universidad Tecnológica de Pereira, por permitirme ver la importancia del desarrollo científico en el desarrollo de la humanidad y el apoyo de unas buenas bases humanitarias para el crecimiento mental y espiritual de un ingeniero.

- A mi director Ph.D Mauricio Granada Echeverri, por su gran confianza y apoyo en el desarrollo de mi proyecto de grado. A él un gran respeto y aprecio por toda la ayuda proporcionada en el tiempo que trabajamos juntos.

- A mis padres, Antonio Escobar por mostrarme el camino indicado en la vida y por mostrarme el gran mundo de la investigación y Suany Vargas por ser mi apoyo incondicional y mi esperanza. Gracias por ser los mejores profesores que un hijo puede tener: A ambos un profundo respeto, amor y admiración.

- A mis hermanos Laura Mónica y Juan Pablo, por su compañía incondicional y constante apoyo, como siempre juntos en todo momento. A ellos un gran amor y aprecio.

- A mi prometida Laura, por ser mi mejor amiga y luz de inspiración en cada paso que damos juntos. Gracias por creer en mí y permitirme compartir las mejores experiencias de mi vida. A ella todo mi amor y mi más profundo respeto.

- A Gloria Marín, Ana Isabel y Anadelia Berrio por su compañía y apoyo.

- A mis compañeros de pregrado, en especial a Carlos Gonzales, Alejandro Suarez y Manuel Pineda por sus consejos y su compañía en las mejores facetas de la universidad. A ellos mis agradecimientos.

- A mis compañeros Andrés Domínguez, Luis Miguel Escobar y David Álvarez por brindarme su apoyo incondicional y su amistad en este proceso.

- A cada uno de los integrantes de los grupos de investigación Planeamiento en Sistemas Eléctricos, Dinop y Gaope.

- A Ramón A. Gallego y Eliana Toro, por creer en mis capacidades e incentivarme a ser mejor cada día. A ambos mi profundo respeto y gran aprecio.

- A Anand Subramanian por su amistad, gran confianza y paciencia en todo este proceso. Agradezco mucho el poder haber cruzado caminos. A él mi profundo respeto y mi aprecio.

- A Frederico Gadelha Guimaraes por sus consejos y apoyo en el poco tiempo que pudimos trabajar juntos. A él mi aprecio y profundo respeto.

- A Claudio Contardo por su apoyo incondicional y su gran confianza en este proceso. Una gran persona y excelente profesional. A él un profundo respeto y aprecio.

- A mi abuela María Edilma Henao, por mostrarme la mejor cara de la vida y por su eterna compañía en las etapas más importantes de la misma. A ella un eterno amor, aprecio y agradecimiento por hacerme el ser humano que soy hoy en día. Juntos en la distancia.

RESUMEN

En este trabajo se presenta una metodología híbrida para resolver el problema de ruteo de vehículos capacitado o CVRP. Se plantea una propuesta para resolver un problema CVRP para un único depósito que utiliza en su interior la solución del modelo del agente viajero (TSP) que se deriva del modelo del problema VRP con distancias simétricas y que no incluye el depósito. El problema resultante se resuelve usando un algoritmo genético especializado de Chu-Beasley que utiliza como etapa de mejoría un algoritmo exacto de enumeración implícita 0-1 de Balas. El algoritmo genético contiene en cada iteración una población de individuos que representan alternativas de solución separadas en rutas. Estas son codificadas de forma eficiente usando un solo vector de tamaño constante. Dentro del algoritmo genético se crean rutas nuevas, se eliminan rutas existentes, se intercambian clientes de una misma ruta y se intercambian clientes que pertenecen a diferentes rutas. Una vez finalizado el ciclo selección-recombinación-mutación, el descendiente generado representa una nueva propuesta de solución con un conjunto de rutas y de clientes por ruta. En este momento se inicia la etapa de mejoría local, la cual toma cada ruta del individuo por separado y la optimiza usando un algoritmo exacto de enumeración implícita 1-0 de Balas. En esta etapa no se crean ni se eliminan rutas del descendiente.

El algoritmo genético parte de una población inicial de buena calidad construida usando la combinación de varias técnicas heurísticas, dentro de las cuales se encuentra el algoritmo de Lin-Kernighan. Se implementa también un algoritmo heurístico que usa el concepto del vecino más cercano y un algoritmo de ahorros modificado. Para el algoritmo genético se propone el uso de dos métodos de recombinación, para la generación de descendientes, los cuales se pueden usar de manera individual o de manera coordinada. Para el algoritmo genético también se presenta una etapa de mutación aleatoria-controlada y una etapa de mejoría que utiliza un algoritmo basado en la técnica de enumeración implícita 0-1 de Balas, el cual hace las veces de un algoritmo de ordenamiento intra-ruta exhaustivo. En el algoritmo de enumeración 1-0 de Balas implementado se resuelve cada vez un problema TSP relajado, es decir, sin restricciones de *sub-tours*, se identifican los *sub-tours* que pueden surgir, se adicionan únicamente las restricciones de *sub-tours* violadas y se resuelve de nuevo el TSP hasta que no existan *sub-tours*. En consecuencia, las rutas individuales del descendiente, entregadas por el algoritmo genético, son optimizadas utilizando un método que no requiere la solución de subproblemas de programación lineal. Los individuos mejorados pueden entonces reemplazar a algún individuo de la población inicial si tienen mejor función objetivo y cumplen diversidad.

Para la validación de la propuesta, se emplean varias instancias de prueba disponibles en la literatura especializada, con tamaños entre 32 y 80 clientes.

Los archivos de datos de estas instancias han sido modificados y adaptados, sin alterar su información, para una ágil lectura por parte de los algoritmos.

TABLA DE CONTENIDO

CAPÍTULO 1

INTRODUCCIÓN

1.1 Justificación:

La optimización de ruteo de vehículos capacitado (CVRP) es un nombre genérico que hace referencia a una gran variedad de problemas en donde un grupo de clientes deben ser atendidos por una flota de vehículos que inician su recorrido o ruta en una bodega principal o depósito, visitan los clientes asignados en su ruta una sola vez y regresan de nuevo al depósito donde iniciaron su recorrido. El problema consiste en establecer las posibles vías que han de representar las rutas de costo mínimo por donde se atenderán las demandas de dichos clientes, los cuales se encuentran geográficamente dispersos y están asociados a una o varias restricciones que deben ser satisfechas. Esta forma de definir el problema se deriva de su primera formulación, planteada hace más de 40 años, y ha dado origen a una gran cantidad de variantes del problema.

En su formulación más básica, los clientes deben ser visitados una sola vez y los vehículos deben salir del depósito, visitar los clientes asignados y regresar al depósito.

La figura 1.1 muestra un ejemplo para un depósito, que se simboliza con el número cero, y ocho clientes, nombrados de 1 a 8. En este esquema se representa una propuesta de solución, que puede o no ser la óptima, y en la cual se establecen tres rutas para tres vehículos. Puede observarse que algunos vehículos visitan más clientes que otros, ya que esta condición no hace parte del problema básico. La figura muestra los clientes que debe visitar cada vehículo y el orden en que debe visitarlos ya que los caminos entre clientes están orientados.

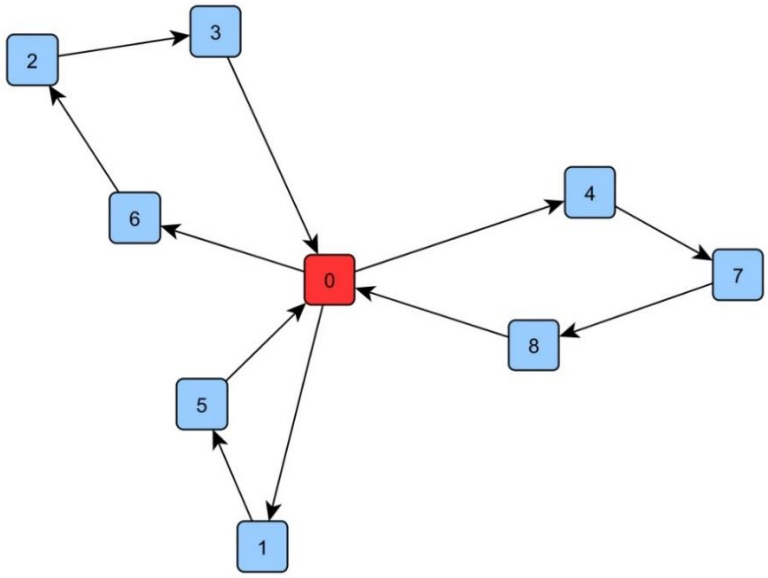


Figura 1.1. Representación gráfica VRP

El problema del ruteamiento de vehículos es uno de los problemas de optimización combinatorial más estudiados debido a su impacto en una gran variedad de problemas de la vida real y a su complejidad matemática. También es un problema de alto interés en el área de la investigación de operaciones para probar nuevas metodologías de solución de problemas NP-completos. El crecimiento acelerado de la población mundial nos cuestiona sobre qué tan preparados están los países y sus organizaciones para afrontar el abastecimiento y el desplazamiento de la sociedad en el futuro próximo. Se considera que en la actualidad aproximadamente 8 millones de consumidores son visitados diariamente por una variada flota de vehículos de carga que manejan en promedio 15.6 millones de paquetes, por lo que el control y seguimiento de la evolución de dichos sistemas necesita hoy en día de un trabajo investigativo constante, el cual puede resultar insuficiente en la actualidad. Un aspecto importante es que las universidades son las primeras llamadas a desarrollar conocimiento matemático que permita avanzar en la

solución de estos problemas y es necesario apuntar al desarrollo de metodologías universales aplicables a una gran variedad de problemas.

Históricamente se ha entendido el VRP como una generalización del TSP. Tanto el problema VRP como el TSP poseen gran complejidad computacional y se enmarcan en la categoría NP-completos, lo que quiere decir que hasta el día de hoy no se ha encontrado un algoritmo que pueda resolverlos en tiempo polinomial.

Con respecto a lo anterior, se puede mostrar fácilmente que la búsqueda exhaustiva de la solución del problema TSP con *n* clientes requiere evaluar un conjunto de *(n-1)!* posibles soluciones. En la actualidad, las máquinas de cómputo existentes resultan inadecuadas para encontrar la solución de problemas con un número medianamente grande de clientes, realizando una búsqueda exhaustiva. Suponiendo que los problemas de la vida real trabajan para resolver instancias de alrededor de 20 clientes, resulta aun computacionalmente imposible dar solución a dichos problemas usando técnicas de búsqueda exhaustiva dado que se requiere explorar un espacio aproximado de 2.43×10^{18} posibles soluciones (en el caso de 21 clientes). Si suponemos que la evaluación de cada propuesta requiere de un tiempo de 1×10^{-9} segundos, la solución óptima se encontraría en aproximadamente 83.81 años, por lo que encontrar soluciones a corto plazo resulta ser una tarea imposible usando enumeración explícita. Si consideramos que los problemas de la vida real tienen generalmente más de 20 clientes y que el espacio de soluciones crece exponencialmente con el crecimiento del número de clientes, el tiempo se vuelve aún más prohibitivo. Usando técnicas como la enumeración implícita es posible resolver problemas de 100 clientes en horas y en ocasiones hasta en minutos, si se combinan adecuadamente con inicializadores heurísticos y técnicas metaheurísticas como los algoritmos genéticos.

La principal motivación para usar la técnica de enumeración implícita 1-0 es que, a diferencia de técnicas de exploración implícita como *Branch and Bound*, *Branch and Cut* o *Branch and Cut and Price*, en el algoritmo de Balas no se requiere resolver problemas de programación lineal (PL), lo que lo hace más eficiente desde el punto de vista computacional. En [19] se muestra una aplicación de este algoritmo al problema de planeamiento de sistemas de transmisión en sistemas de gran tamaño y complejidad de la literatura especializada, con muy buenos resultados. Los autores reportan un buen desempeño del método.

1.2 Antecedentes y Estado del Arte

El problema general de ruteamiento de vehículos (VRP) es uno de los problemas de optimización más interesantes del área de investigación de operaciones que ha sido analizando y estudiando ampliamente desde su primera aparición en la literatura por medio de la formulación aplicada a la distribución de combustible realizada por Dantzig y Ramser en 1959 [1], [5]. A partir de este trabajo varios autores han concentrado sus esfuerzos en la búsqueda de técnicas y modelos eficientes que permitieran dar solución a este problema. Entre los autores se encuentra la primera referencia del TSP múltiple o *m*-TSP propuesto por Miller, Tucker y Zemlin en 1960 [24], el cual es una generalización del TSP básico y que agrega un depósito y *n* agentes viajeros. Para este problema se planteó como objetivo principal la construcción de *n* rutas que correspondieran a un agente viajero diferente encargado de recorrerlas, de modo que cada una de las ciudades o clientes fuera visitado una única vez por un único agente o vehículo y en donde cada segmento del recorrido o ruta, debía iniciar y finalizar en un depósito existente.

A partir de la formulación del TSP, se desarrollaron adaptaciones del *m*-TSP con variantes. Trabajos como *The Vehicle Routing Problem* [22] propuesto por Toth y Vigo, son tomados como referencias obligadas cuando se estudia el problema del VRP y su modelamiento. A partir de estas formulaciones y al apoyo científico de diferentes grupos de investigadores interesados en el problema, es posible encontrar en la última década importantes avances en algoritmos y métodos de optimización eficientes que poseen la capacidad de dar solución a problemas básicos del VRP de gran tamaño y complejidad. Adicional al avance teórico, y paralelo a esto, hoy en día es posible ver plasmado el desarrollo de innovaciones algorítmicas y tecnológicas, que han permito conectar, problemas del papel a la realidad por medio tecnologías como el GPS, herramientas de posicionamiento local y/o global y software de reconocimiento de movimiento, como es posible ver en algunos de los medios de transporte usados en la actualidad.

Estos tipos de problemas a su vez poseen otras variantes que consideran aspectos característicos como la inclusión de escenarios, características físicas del problema, aspectos ambientales, armonización de la longitud de las rutas para equilibrar el esfuerzo de los conductores, armonización de las cargas de los vehículos y costos de combustibles, entre otras. Para el problema VRP existe una amplia variedad de bibliografía que muestra su desarrollo, autores como Bodin,1975 [3], presentan una estructura taxonómica para el problema VRP sin considerar demandas reales y para los problemas de programación, agregando a su vez dificultades computacionales que se pueden encontrar en la implementación de soluciones para los mismos. Esta primera parte es la diferencia entre el problema de ruteamiento y el problema de programación, se indica que una ruta de un vehículo es una secuencia de entregas y recogidas

en las cuales el vehículo debe seguir un orden, empezando y terminando en un depósito. La programación de un vehículo es una secuencia de entregas y recogidas agrupadas asociado a un conjunto de llegadas y salidas del depósito un cierto número de veces, que corresponde al número de rutas que debe realizar el mismo vehículo en un límite de tiempo específico. Las características que fueron tomadas en cuenta para realizar la clasificación fueron las siguientes: clasificación de la red, número de vehículos que deben ser ruteados, análisis de un vehículo VRP, técnicas de solución, el problema de ruteamiento de múltiples vehículos y los problemas encontrados en la implementación computacional. Bodin y Golden (1981) [3], [13], presentan una clasificación del problema de ruteamiento de vehículos y el problema de programación. Nuevamente se explica la diferencia entre el problema de ruteamiento y el problema de asignación, presentan una taxonomía del problema combinado de ruteamiento y asignación que según su apreciación se acerca más a los problemas reales. Presentan estrategias de solución actuales y muestran la jerarquía de los problemas de programación desde los más simples hasta los más complejos, además describen tres posibles combinaciones de los problemas de ruteo y programación. En la clasificación proponen como características principales: tiempo de servicio de un nodo particular o arco, número de depósitos, tamaño y tipo de la flota, naturaleza y localización de la demanda, red subyacente, restricción de capacidad de vehículos, tiempo máximo de rutas, costos, operación y objetivos. También presenta una clasificación de las estrategias de solución.

En el estudio de Laporte y Norbert (1987) [15], se presenta una clasificación de los métodos exactos para resolver el problema de ruteamiento de vehículos y los clasifica en tres categorías *i)* métodos directos de búsqueda en árbol, *ii*) Programación dinámica y *iii*) Programación lineal entera.

Fusaka (2004) [7] presenta un modelo lineal entero para describir el problema del CVRP. En la parte final del documento se hace una revisión en términos generales de los algoritmos para resolver el problema de VRP haciendo referencia a técnicas exactas, técnicas heurísticas y técnicas metaheurísticas.

Finalmente en otras publicaciones como en [20] es posible observar la tendencia por una combinación entre varias técnicas de optimización conocidas, sobre todo de aquellas que no requieren mucho tiempo computacional y alcanzan resultados de muy buena calidad. En otras áreas de la investigación dentro del campo de la optimización matemática se observa una tendencia respecto al uso de diferentes técnicas, optando cada vez más por la diversidad metódica y algorítmica que pueda resolver problemas específicos o resultar en metodologías generales que puedan ser utilizadas en cualquier área de la optimización.

1.3 Objetivos Generales y Específicos

1.3.1 Objetivo General

Desarrollar una metodología de solución híbrida que permita planificar rutas de mínimo costo para el problema de ruteamiento de vehículos capacitado CVRP, disminuyendo el tamaño del espacio de búsqueda.

1.3.2 Objetivos Específicos

- Estudiar los problemas más significativos y actuales relacionados con el problema de ruteamiento de vehículos capacitados para un solo depósito.
- Definir los modelos matemáticos que representen mejor el problema del ruteamiento de vehículos capacitado con un único deposito (CVRP).
- Elaborar una metodología que permita implementar el modelo matemático del problema de optimización combinando técnicas heurísticas, metaheurísticas y exactas.
- Implementar los algoritmos en lenguaje C++ y desarrollar los *solvers* necesarios.
- Probar la metodología propuesta en instancias disponibles en la literatura especializada.

1.4 Aportes del proyecto

El desarrollo de la investigación tiene como aporte los siguientes aspectos:

- Se presenta una propuesta basada en población diferente a la tendencia de uso de algoritmos de trayectoria, propuestos en literatura, cuando se involucran técnicas metaheurísticas para la búsqueda de soluciones óptimas o cuasióptimas. En este trabajo se propone el uso de un algoritmo genético del tipo Chu-Beasley como problema maestro y un algoritmo de enumeración implícita 1-0 como problema esclavo. El algoritmo genético comienza con una población inicial construida usando técnicas heurísticas, realiza propuestas de rutas para los vehículos,

verifica factibilidad y recombina y muta las rutas actuales. Antes de determinar si los descendientes pueden reemplazar a algunos individuos de la población actual se realiza una etapa de mejoría exhaustiva para las rutas individuales, la cual consiste en usar un algoritmo de enumeración implícita 1-0 de Balas sin restricciones de *sub-tours* al cual se van adicionando las restricciones de *sub-tours* violadas en caso de ser requeridas. Al ser las rutas subconjuntos de clientes, el problema exacto resulta de menor tamaño y complejidad que el problema completo.

- El método híbrido propuesto para resolver el problema de ruteamiento de vehículos con capacidad limitada (CVRP), involucra un conjunto de procedimientos heurísticos para la generación de la población inicial en conjunto con un procedimiento de rotación para agregar diversidad a la población.

- Se aplican dos métodos de recombinación, usados de forma alternativa, de tal forma que la población mantenga su diversidad y no pierda características importantes de sus predecesores.

- Como etapa de mutación, se implementó un método de intercambio simple entre las diferentes rutas existentes.

- La etapa de mejoría realiza el ordenamiento óptimo de visita de los clientes de cada ruta de un descendiente de la población por medio de la solución de un modelo reducido del problema del agente viajero (TSP) que incluye al depósito. Se utiliza el método de enumeración implícita 0-1 de Balas porque no requiere resolver problemas de programación lineal.

- Se incluyen dos subrutinas que permiten identificar *sub-tours* en las rutas individuales, y construir la restricción de *sub-tour* requerida para adicionarla al modelo.

1.5 Estructura del documento

Este trabajo está organizado de la siguiente manera:

- En el capítulo 2 se presenta la descripción y formulación matemática del problema del problema de ruteo de vehículos capacitado CVRP, su función objetivo y sus variables de decisión.

- En el capítulo 3 se propone una metodología basada en un algoritmo genético tipo Chu-Beasley [4] modificado para el problema maestro y un método de enumeración implícita 0-1 para el problema esclavo que realiza la etapa de mejoría local.

- En el capítulo 4 se muestra la aplicación del algoritmo en diferentes instancias disponibles en la literatura especializada y los resultados obtenidos.

- En el capítulo 5 se presentan las conclusiones del trabajo respecto a la metodología propuesta y los resultados obtenidos, además de algunas recomendaciones y propuestas para proyectos futuros.

CAPÍTULO 2

DESCRIPCIÓN DEL MODELO MATEMÁTICO

El problema de ruteamiento de vehículos (VRP) es presentado en su forma más simple, como un problema que busca diseñar un conjunto de rutas que permitan atender todas y cada una de las demandas presentadas por un conjunto de clientes usando para esto una flota homogénea de vehículos de carga limitada que deben salir y regresar de un deposito principal. Dentro del problema cada cliente se encuentra caracterizado por un conjunto de atributos tales como una demanda específica y una posición en el espacio. Los clientes se encuentran dispersos geográficamente dentro de una región limitada. Dado que este problema ha sido ampliamente estudiado por más de 40 años, desde su primera formulación, un grupo de investigadores ha manifestado su interés por analizar el impacto económico que implica su aplicación en diversos problemas de la vida real. En áreas como el transporte de carga, de alimentos, de recursos y de desechos, resulta fundamental para el funcionamiento eficiente de las empresas. También es muy útil para el transporte de personas dentro de una ciudad y entre ciudades. Todo esto independientemente de si se utilizan vehículos, embarcaciones, aviones o trenes. Debido a la gran variedad de problemas donde se puede aplicar, el propio problema de VRP ha evolucionado en los objetivos que se persiguen y en las restricciones que se consideran. La intención es aproximar cada vez más el problema teórico a los problemas de la vida real.

Una de las variables más conocidas del VRP es el CVRP o problema de ruteamiento de vehículos capacitado, que parte básicamente del mismo planteamiento, con la particularidad de que los vehículos tienen una capacidad específica y limitada.

Una manera formal de definir el problema del CVRP es la siguiente. Sea $G = (V, E)$ un grafo completo con un conjunto de vértices $V = \{0,..., n\}$, donde el vértice 0 representa el depósito y los demás nodos representan a los diferentes clientes del problema. Sea el conjunto de arcos E las posibles conexiones encargadas de unir cada uno de los clientes con otros clientes y con el depósito. Cada arco $\{i, j\} \in E$ tienen costos positivos asociados c_{ij} y cada cliente $i \in V' = V\backslash\{0\}$ posee una demanda asociada d_i. Sea $C = = \{1,..., m\}$ el conjunto de vehículos con capacidad homogénea Q. El CVRP consiste en construir un grupo de máximo m rutas en donde se cumpla que: (i) cada ruta debe iniciar y finalizar en el deposito; (ii) todas las demandas de los clientes deben ser satisfechas; (iii) la capacidad del vehículo no debe sobrepasar el límite de capacidad Q; (iv) cada cliente es debe ser visitado una única vez por un solo vehículo.

A partir de estos parámetros definidos por en el CVRP, el objetivo de este problema consiste en minimizar la distancia recorrida por el conjunto de rutas que visitan a todos los clientes sin exceder la capacidad de los vehículos que puede encontrarse limitado por el problema o con una flota homogénea infinita.

2.1 Formulación matemática

En esta sección se presentan los modelos usados en el desarrollo del problema: el modelo del CVRP básico usado para la asignación de clientes a cada ruta y el modelo matemático del TSP, usado en el ordenamiento óptimo de los clientes en cada una de las rutas del problema.

2.1.1 Notación

A. Conjuntos

V Conjunto de vértices del sistema

S Conjunto de consumidores

B. Parámetros

C_{ij} Costo asociado a la activación de un arco (*i,j*)

K Número de rutas existentes

$r(S)$ Número de arcos

n Número de vertices

C. Variables

x_{ij} Variable de decisión. Representa la conexión entre dos vértices (*i,j*)

2.1.2 Modelo CVRP

El problema de ruteamiento de vehículos capacitado está planteado como un problema de optimización cuya función objetivo está asociada a la minimización del costo o distancia total recorrida por todos los vehículos, mientras se satisfacen restricciones asociadas a la visita de cada cliente por parte de un único vehículo, de tal forma que se atiendan todos los clientes y se cumpla la restricción de capacidad del vehículo. El siguiente modelo presentado, corresponde al ACVRP propuesto en Toth y Vigo [22]. En este caso las variables x_{ij} pueden asumir el valor de 1 si el arco (*i,j*) hace parte de la solución del problema y 0 en caso contrario.

La forma en que se encuentra propuesto el modelo es extensamente utilizado tanto para el problema del VRP con distancias simétricas como para el VRP con distancias asimétricas.

$$\min \sum_{i \in V} \sum_{j \in V} C_{ij} x_{ij} \tag{2.1}$$

$Sujeto\ a$:

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

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\} \tag{2.3}$$

$$\sum_{i \in V} x_{i0} = K \tag{2.4}$$

$$\sum_{j \in V} x_{0j} = K \tag{2.5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq 0 \tag{2.6}$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \in V \tag{2.7}$$

La ecuación (2.1) representa el costo total que debe ser minimizado como meta del problema. Este valor varía en función de las distancias involucradas en cada propuesta particular y que permiten conectar cada grupo de clientes o ruta con el depósito. Las ecuaciones (2.2) y (2.3) garantizan que todos y cada uno de los clientes sea visitado una única vez. Mientras que (2.4) y (2.5) permiten asegurar en al nodo *0*, que en este caso representa el depósito, debe llegar un enlace de entrada y de salida por cada una de las rutas *K* que se encuentran conectadas a él. La ecuación (2.6) aseguran que las conexiones entre todos los clientes no generen *sub-tours* en las rutas, de manera que las rutas formen un ciclo cerrado con el depósito. Y la condición (2.7) define las variables de conexión x_{ij} como binarias, es decir, que sólo pueden asumir el valor 0 o el valor 1.

2.1.3 Modelo TSP

El modelo del agente viajero, usado para resolver las rutas individuales, considera que cada uno de los clientes debe ser visitado una sola vez y no considera la capacidad del vehículo ni la conexión entre el cliente y el depósito existente, tal como lo hace el modelo del CVRP. El TSP busca minimizar la distancia total recorrida entre todos los clientes o vértices del problema, por un único agente viajero. Dada la tendencia a la creación de *sub-tours* entre clientes de una ruta, se consideran estas restricciones únicamente si se detectan *sub-tours*.

El siguiente modelo, es una adaptación del modelo del ACVRP simétrico reducido propuesto por Toth y Vigo en [22]. Seleccionado por su menor tamaño en número de variables.

$$\min \sum_{i \in V} \sum_{j \in V} C_{ij} x_{ij} \tag{2.8}$$

$Sujeto\ a$:

$$\sum_{h<i}^{n} x_{hi} + \sum_{j>i}^{n} x_{ij} = 2 \qquad \forall j \in V \tag{2.9}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{2.10}$$

La ecuación (2.8) representa el costo total que corresponde a la suma de las distancias que deben recorrer los vehículos al realizar las rutas. Esta distancia acumulada debe ser minimizada. La ecuación (2.9) cumple la función de (2.2) y (2.3) para el modelo anterior, pues es un formulación reducida que asegura que cada cliente sea visitado una vez mientras que la condición (2.10) limita la variabilidad de x_{ij} a valores 0 ó 1.

Dado que las restricciones de *sub-tours* implican la creación de una enorme cantidad de inecuaciones para el método de enumeración implícita 0-1, se optó por implementar una subrutina que identifica *sub-tours* dentro de cada ruta, y de ser necesario crea y adiciona restricciones específicas para eliminarlos, por lo que no es necesario considerar las innumerables restricciones de sub-tours asociadas a cada ruta.

CAPÍTULO 3

METODOLOGÍA: ALGORITMO GENÉTICO DE CHU-BEASLEY

3.1 Generalidades

3.1.1 Algoritmo genético

Los algoritmos genéticos son actualmente una de las líneas más prometedoras en el área de la inteligencia artificial la cual hace parte de las denominadas técnicas evolutivas, y fueron originalmente propuestos por John Henry Holland en los años 70 [14]. Esta técnica tiene como estructura básica partir de una población inicial, realizar un proceso de selección, cruzamiento o recombinación, una etapa de mutación y un paso de reemplazo en la población inicial, obteniendo como resultado global la evolución de la población [8], [21].

Para una aproximación computacional, los algoritmos genéticos simulan un proceso de selección natural [8] para dar solución a problemas del campo de la optimización matemática. Por tal motivo, se parte de imitar la naturaleza al simular el ciclo de vida de una serie de individuos que salen de una población inicial, en donde el problema principal hace las veces de medio ambiente y en el cual la función objetivo simula el aspecto complicante del medio ambiente. Los individuos que forman la población son parte de un conjunto de soluciones candidatas al problema. Sólo los individuos mejor adaptados sobreviven a los aspectos complicantes del medio ambiente y pueden pasar sus características

a sus descendientes a través de sus genes. Es importante aclarar que lo que pasa de generación en generación no son los individuos sino los esquemas o combinaciones de genes que permiten a estos individuos adaptarse mejor al medio ambiente. Se realiza una representación abstracta de cada individuo que a su vez representa una solución numérica o alternativa que pasará por todo el proceso evolutivo. Al finalizar este proceso evolutivo se encuentran individuos mejor adaptados al medio ambiente que los que se tenían en la población inicial, esto porque a través del proceso de recombinación se privilegia la propagación de esquemas o conjuntos de genes de excelente calidad mientras se favorece la eliminación de esquemas de baja calidad. Sin embargo esto no asegura que la mejor solución encontrada en la población final corresponda a la solución óptima del problema, aunque existe la posibilidad de encontrarla.

El algoritmo genético del tipo Chu-Beasley pertenece al grupo de las denominadas técnicas metaheurísticas que a su vez consisten en un conjunto de procedimientos evolutivos modificados, basados en el planteamiento básico del algoritmo genético propuesto Holland. La versión de Chu-Beasley presenta una serie de variaciones que hacen de la técnica un método más eficiente y competitivo cuando se resuelven problemas de mediano y gran tamaño y complejidad. Las técnicas metaheurísticas, en general, no requieren de los recursos computacionales (memoria y tiempo) que si exigen las técnicas exactas, las cuales pueden llegar a consumir una gran cantidad de recursos de máquina. Entre las características más importantes de un algoritmo genético tipo Chu-Beasley se destacan las siguientes:

- El uso de la función objetivo separada de la infactibilidad. Cada individuo de la población se califica entonces por estos dos aspectos por separado sin necesidad de usar factores de penalización.

- La infactibilidad puede ser manejada de forma simple, esto permite agregar diversidad a la población y permite evaluar subespacios circundantes más amplios. En ocasiones el manejo de infactibilidad permite una rápida convergencia a la solución ya que habilita recorridos que pueden ser más cortos entre los subespacios actuales y los subespacios donde se encuentra la solución óptima del problema.

- Varía el proceso de recombinación y reemplazo al limitarlo a la generación de un solo descendiente en cada ciclo generacional.

- Es un método de naturaleza elitista, ya que su lógica solo permite el ingreso de un individuo a la población únicamente si este es de mejor calidad que el peor de los individuos existentes.

- Cada individuo que ingresa a la población debe cumplir con un criterio de diversidad, lo que permite explorar subespacios más diversos y ayuda a evitar la convergencia prematura del método a óptimos locales.

- Posee un criterio de aspiración, que permite que un individuo pueda ingresar a la población sin cumplir el criterio de diversidad, siempre y cuando tenga una calidad superior a la incumbente.

- Cuenta con una etapa de mejoría local, la cual se aplica al terminar la ejecución de los mecanismos básicos del algoritmo genético como la selección, la recombinación y la mutación. Esto permite que el individuo resultante tenga la posibilidad de mejorar su calidad. Esta etapa de mejoría generalmente explora regiones vecinas a la solución descendiente para determinar si en este vecindario existen soluciones de mejor calidad. En este trabajo se propone una etapa de mejoría local exhaustiva para cada ruta, es decir, que explore todo el vecindario de cada ruta actual. Es local porque no permite intercambiar clientes de rutas diferentes.

3.1.2 Método de enumeración implícita 0-1 de Balas.

En el campo de la optimización matemática existen una gran cantidad de metodologías que a lo largo del tiempo han intentado llegar a una respuesta óptima de un problema o en su defecto, a la más cercana a la misma, partiendo de las técnicas heurísticas que actualmente son las de más bajo nivel y mayor velocidad computacional por su bajo consumo de recursos, hasta las técnicas metaheurísticas, que generalmente son un conjunto de métodos basados en comportamientos de la naturaleza o bio-inspirados con adaptaciones que permitan generalizar cada método con el fin de ser fácilmente adaptable a cualquier problema contenido en la optimización. Esta característica los ha posicionado como una de las líneas más atractivas en la optimización, por su relativa sencillez de implementación y sus buenos resultados con un mediano uso de recursos de cómputo. Por otro lado, existen problemas en los que es fundamental encontrar un punto de solución óptimo o próximo al mismo, dado que estos pueden ser usados como puntos de comparación para la calibración de métodos heurísticos y metaheurísticos, lo que hace importante el uso de técnicas exhaustivas que se encarguen de alcanzar y superar resultados que sean posibles de ser implementados sobre máquinas de cómputo convencionales. Estos métodos dedicados, actualmente reciben el nombre de técnicas exactas, y a diferencia de las heurísticas y metaheurísticas, estas aseguran el alcance de óptimos de alta calidad pero con un costo computacional bastante elevado.

Estas técnicas exactas han despertado un gran interés en las comunidades científicas por su convergencia en óptimos globales en problemas de pequeño y mediano tamaño, pero para pruebas de sistemas con gran número de variables, se tiene un espacio solución demasiado extenso como para ser explorado eficientemente en una máquina sin las características de una supercomputadora. Con el fin de reducir el espacio de solución en la búsqueda de la solución óptima, existen técnicas exactas adaptadas a un conjunto de técnicas capaces de usar métodos inteligentes para encontrar soluciones de muy buena calidad en tiempos computacionales razonables en comparación con las búsquedas exhaustivas. Estos métodos exactos poseen una gran variedad de formas, las cuales pueden ser modificadas para satisfacer las necesidades de cada tipo de problema específico. Estas técnicas pueden ser agrupadas según el tipo de problema que resuelven en: programación lineal, programación lineal entera, programación binaria, programación lineal entera-mixta, programación no lineal y programación estocástica.

En la actualidad existe una serie de métodos exactos de alto interés en la comunidad científica por su capacidad para resolver problemas de programación lineal entera, alcanzando buenos resultados en problemas de gran tamaño y complejidad. Estos métodos se denominan *Branch & Cut*, *Branch & Price* y *Branch & Cut & Price* [18]. Actualmente estas técnicas están siendo extensamente usadas en diferentes áreas de la investigación de operaciones.

Todos estos métodos normalmente están compuestos internamente por árbol de posibles soluciones del problema, expandido por medio de cuadros simplex y controlado con el uso de técnicas especializadas que evitan la exploración exhaustiva de todo el árbol de solución. Sin embargo, con la tendencia hacia la solución de problemas más cercanos a la vida real, se encuentran problemas con una gran cantidad de características que hacen que el uso de recursos, como el tiempo de procesamiento, sea elevado, al punto que en ocasiones algunos algoritmos deben ser divididos para una ejecución paralela de los sub-problemas que lo componen.

Observando este inconveniente hace aproximadamente 40 años los científicos que trabajaban en el campo de la investigación matemática y el área de la optimización, identificaron la incapacidad computacional de la época en la búsqueda de soluciones para problemas con una gran cantidad de variables. Entre estas investigaciones, surgió la propuesta de un método contenido en la línea de la programación binaria el cual se basa en evitar el uso de cuadros simplex, reemplazándolo por un conjunto de pruebas computacionalmente livianas que se encargan de determinar las variables activas o básicas que deben hacer parte de la solución final del problema.

Esta propuesta fue planteada en [2], recibiendo el nombre de enumeración implícita 0-1 o comúnmente encontrado en la literatura como método de Balas, en mención a su creador Egon Balas, el cual propuso dos metodologías que se encargan de identificar posibles cortes en el árbol de solución para soluciones

que no fueran promisorias para ser analizadas. Estas técnicas se dividen en métodos de expansión y métodos de sondaje, las cuales seleccionan las mejores variables para ser analizadas y en los mejores casos, determinan desde cualquier estado del método, que caminos deben ser evitados con respecto a la variable expandida.

Con base en el planteamiento básico del algoritmo, un grupo de matemáticos dedicaron sus esfuerzos al perfeccionamiento de estas técnicas de expansión y de sondeo. Sin embargo, el acelerado crecimiento de los métodos basados en la filosofía *Branch & Bound* han dejado el planteamiento de la enumeración implícita 0-1 como un algoritmo poco interesante en comparación con las implementaciones ya existentes de técnicas basadas en el uso de cuadros simplex.

3.2 Descripción del algoritmo propuesto

En esta sección se propone un algoritmo genético de Chu-Beasley [8] modificado y combinado con un conjunto de técnicas heurísticas y exactas que permita dar una solución al problema de ruteo de vehículos capacitado (CVRP) usando la variante que trabaja con flota homogénea y no cuenta con un número limitado de vehículos. Para esto, el algoritmo genético debe comenzar con la generación de una población inicial la cual se construye usando procedimientos heurísticos capaces de generar individuos diversos y de buena calidad por medio del ordenamiento de clientes sin el depósito.

Para el proceso de selección, se realizan dos torneos entre un número variable de individuos seleccionados aleatoriamente de la población inicial, los cuales otorgan a los ganadores para la siguiente etapa. En la recombinación se usa un método de recombinación por rutas adaptado para el CVRP. El descendiente resultante, pasa por una etapa de mutación simple realizada variablemente, para después ser divido en las diferentes rutas que lo componen por medio de la capacidad del vehículo del problema. Como etapa de mejoramiento se usa el método exacto de enumeración implícita 0-1 para el ordenamiento intra-ruta, el cual fue generalizado y adaptado para la solución del TSP que hace parte de un sub-problema del CVPR. Finalmente el individuo es presentado a la población por medio de los criterios de diversidad y calidad. El proceso se realiza de forma iterativa hasta alcanzar el criterio de parada que fue asignado por un número máximo de iteraciones.

3.3 Población inicial: Proceso heurístico

Para la solución inicial, se determinó el uso de tres heurísticas diferentes capaces de determinar una población diversa y de buena calidad. Para esto se analizaron un conjunto de heurísticas que reportan buenos resultados para el problema del agente viajero (TSP) [12] en la literatura especializada. Se determinó usar las heurísticas: Lin-Kernighan [16], vecino más cercano [8] y un algoritmo de ahorros modificado [20]. Se propone usar estos algoritmos separadamente de manera que el resultado final sea la creación de una población de buena calidad, que ayude a disminuir el espacio de búsqueda del algoritmo. Para esto, se plantea el uso de la siguiente estrategia:

1. Dividir la población inicial de acuerdo a un porcentaje pre-especificado para cada una de las heurísticas.

2. Resolver el problema del agente viajero o TSP para todos los clientes sin incluir el depósito. Figura 3.1.

3. Tomar las repuestas obtenidas por cada una de las heurísticas y agregarlas a la porción de población asignada.

4. Tomar cada individuo de la población y dividirlo en rutas dependiendo de la capacidad limitada del vehículo presentado en la instancia.

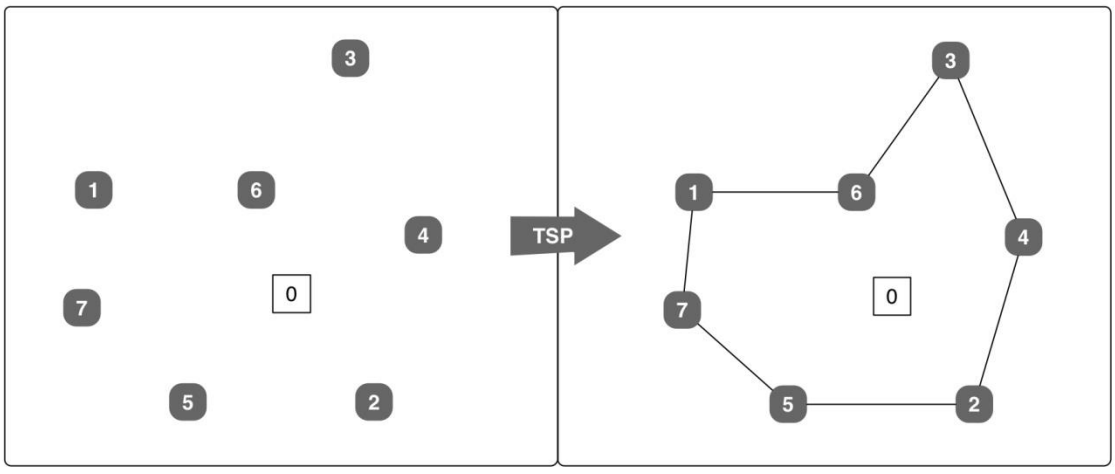


Figura 3.1. Generación del gran *tour*

3.3.1 Generación y división de rutas

Para la generación de la población inicial del algoritmo se planteó el uso de tres técnicas heurísticas capaces de determinar una ruta de bajo costo que uniera a todos los clientes del problema sin incluir el depósito, como se muestra en la figura 3.1. Para esto fue necesaria una búsqueda bibliográfica de las heurísticas con mejores resultados resolviendo el problema TSP, el cual representaría la ruta de menor costo podría ser fácilmente modificado en próximas etapas.

Principalmente, se desarrolló una subrutina que dividiera la población inicial en tres partes diferentes, las cuales pueden ser variadas al inicio del algoritmo para otorgar flexibilidad al método, de manera que sea posible probar las diferentes variaciones entre el tamaño de la población y el tamaño de las fracciones que la componen.

El siguiente paso fue una implementación de un algoritmo constructivo capaz de calcular la ruta mínima entre los clientes y otro que permitiera agregar un factor de aleatoriedad a la población. Con base en esto, fue desarrollado un procedimiento de la heurística del vecino más cercano y un algoritmo de ahorros, propuesto en [20], en donde fue modificado el manejo de la aleatoriedad para otorgar diversidad a la población inicial. Además del uso de la librería Lin-Kernighan-Helsgaun (LKH) para la generación de una ruta mínima de buena calidad.

Para la etapa de llenado la población del algoritmo genético, se propuso tomar cada una de las heurísticas y usarlas de la siguiente manera:

3.3.1.1 Lin-Kernighan-Helsgaun

De acuerdo a la literatura especializada, el algoritmo de Lin-Kernighan [16] es una de las técnicas heurísticas con mejores resultados para el problema del TSP, dado que combina una serie de métodos especializados que resultan ser bastante eficientes en la búsqueda de resultados de buena calidad.

Para el uso de esta técnica, actualmente existe una implementación eficiente en la librería Lin-Kernighan-Helsgaun (LKH), que libera su código en su totalidad a la comunidad de desarrolladores para su uso libre como aplicativo externo o como librería estática. Por tal motivo, en la etapa de construcción de la población inicial se plantea el uso de la librería como un *solver* externo para la generación de una secuencia que representará la ruta mínima del TSP compuesto por los clientes del problema.

Con base en el conocimiento del método, es posible observar que sin importar el punto de inicio, el algoritmo posee una respuesta constante al problema que se resuelve, por lo que debe ser realizado un proceso de rotación de la

secuencia de clientes resultante para la construcción de rutas en la porción de la población asignada.

3.3.1.2 Heurística vecino más cercano

La heurística del vecino más cercano [8], es una de las técnicas con mejor equilibrio entre buenos resultados y tiempo computacional, dado que tiene como objetivo conectar los nodos más cercanos al nodo en el que se encuentra posicionado en ese momento para después repetir el mismo procedimiento con el nodo efectivamente conectado, hasta completar la visita de todos los nodos disponibles. Su principal atractivo como técnica para la construcción de la población inicial está centrado en la esencia del algoritmo, al tener como característica la generación de múltiples individuos diferentes al iniciar el método desde un punto inicial diferente. Para esto se ordenó la secuencia de construcción de la siguiente manera:

1. Asignar un identificador a los clientes con respecto a su posición en la tabla de posiciones de la instancia, tomando como punto de inicio el primer cliente

2. Realizar la búsqueda de los vecinos más cercanos a partir de punto de inicio.

3. Cambiar de punto de inicio por el siguiente cliente en la lista y realizar el paso 2 hasta generar individuos suficientes para llenar la fracción correspondiente en la población inicial.

3.3.1.3 Heurística de ahorros modificado

Como método preventivo para evitar que el algoritmo genético quede atrapado rápidamente en óptimos locales, se implementó un algoritmo de ahorros como el planteado en [6], [17] que fue modificado inspirado en el algoritmo de solución inicial propuesto en [20] con una variante en la sección aleatoria del método, para agregar una gran diversidad a la población.

La especialidad de esta heurística radica en la capacidad de engañar al método de ahorros, al poseer una serie de factores aleatorios capaces de modificar la distancia entre clientes, por lo que el procedimiento toma las distancias desde el punto analizado hasta los demás clientes no visitados y los multiplica por una serie de variables aleatorias que puede hacer que el algoritmo vea algunos puntos más cercanos o más alejados de los que realmente están. Esta técnica de selección va ligada a la capacidad de los clientes de manera que se agreguen clientes que no dañen la restricción de capacidad máxima del vehículo.

Esta heurística se diferencia de los métodos del vecino más cercano y Lin-Kernighan, en que esta es la encargada de llenar el espacio restante en la

población, lo cual permite hacer pruebas del algoritmo genético en donde se tenga posibilidad de tener una gran población con un alto nivel de diversidad.

Después de finalizado el proceso de generación de individuos, se recorre la matriz de la población y cada uno de las secuencias es dividida en rutas que toman como base la capacidad máxima del vehículo. En la figura 3.2, *R* representa la solución del problema *TSP* (orden en que deben ser visitados los clientes), *q* representa la capacidad del vehículo (10) y el vector *Ip* almacena el número del cliente donde se inicia el proceso de separación en rutas. En el ejemplo se inicia en el cliente 1. El resultado del proceso de separación en rutas se almacena en el mismo vector *R* que contiene los clientes ordenados.

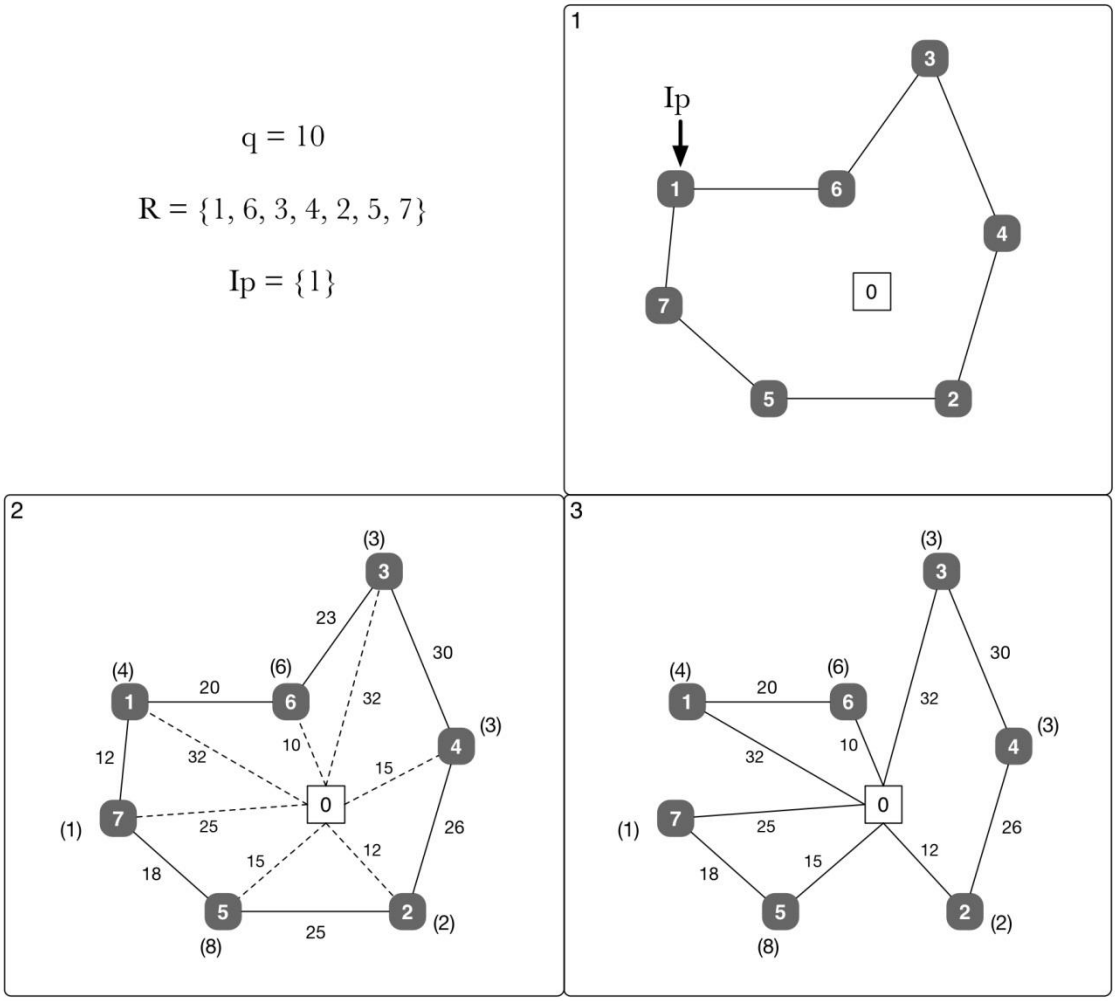


Figura 3.2 División del gran *tour* en rutas.

3.4 Codificación empleada para el CVRP

Después de una revisión de las múltiples codificaciones empleadas para el VRP y sus variantes se resalta el hecho del uso de estructuras de datos

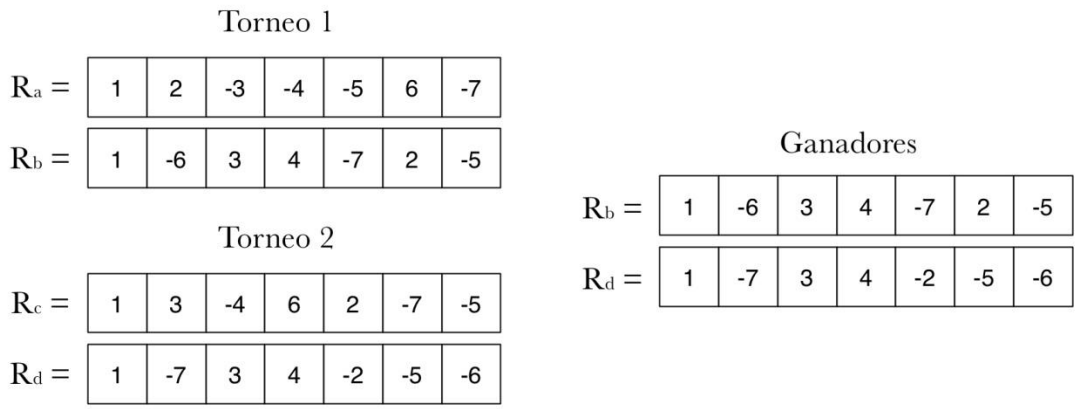
encargadas de separar las rutas por medio de ceros, que representaban el depósito al que llegaba y del que salía cada ruta. Otras codificaciones se encaminan al planteamiento matricial de cada ruta por lo que se podían presentar inconvenientes como: la creación de una matriz con un número variable de rutas en donde el espacio reservado para las rutas debía ser el mismo que el número de depósitos. En el peor de los casos se tiene un grupo de matrices con tamaño variable, lo cual representa un gran gasto computacional en reasignación de memoria en cada una de las matrices de cada estructura, en el proceso de reemplazo o en una simple creación y destrucción de estructuras completas con un bajo uso de reasignación de memoria lo que representa una dificultad importante en problemas de gran tamaño y complejidad.

Una de las propuestas realizadas por este proyecto se enfoca en el uso de una codificación eficiente, que se oriente en la disminución del esfuerzo computacional y en el equilibrio en el uso de la memoria. Para esto, se propone usar un arreglo numérico entero compuesto por el orden de cada una de las rutas separadas por un valor negativo al final de la misma. De esta manera, no sería necesario el uso de memoria dinámica en la reasignación de espacio. Esto permite que la población inicial sea una matriz de enteros, en el caso de varios depósitos, y no una matriz de estructuras.

$$Sol = \begin{cases} R = \{1, -6, 3, 4, -2, 5, -7\} \\ C = \{c_1,\ c_2,\ c_3\} \\ F_{obj} \end{cases}$$

3.5 Selección

Para la selección de individuos de la población, se realizan dos etapas encargadas de definir los dos mejores individuos, resultado de dos torneos que son llevados a cabo con la selección aleatoria de un número variable de padres los cuales entraran a ser comparados por su respectiva función objetivo [8]. El ganador de cada uno de los torneos será reservado con su respectivo identificador en la población para ser llevado a la siguiente etapa del algoritmo.



3.6 Recombinación

En estar parte del trabajo se analizaron varias propuestas de recombinación como la recombinación de un solo punto, la recombinación de dos puntos y la recombinación multi-punto [8]. También se analizaron recombinaciones especializadas donde se intercambian rutas o partes de rutas entre individuos.

Con el fin de preservar atributos de buena calidad presentes en los padres, se utiliza una etapa de recombinación basada en un proceso de combinación de rutas de cada uno de los padres para que el hijo resultante contenga segmentos de recorridos existentes en cada uno de ellos. Para esto, se lleva a cabo el siguiente procedimiento:

a. En el padre 1 seleccionar aleatoriamente una de sus rutas y marcar esta ruta para que no sea seleccionada nuevamente en el futuro.

b. Copiar la ruta seleccionada en el descendiente, verificando que los clientes de la ruta seleccionada no se encuentren ya en el descendiente. Si ya existen eliminar este cliente de la ruta que se va a adicionar. Si todos los clientes se encuentran en el descendiente finalizar.

c. Al digito marcador de fin de ruta cambiarle el signo negativo por positivo.

d. En el padre 2 seleccionar aleatoriamente una de sus rutas y marcar esta ruta para que no sea seleccionada nuevamente en el futuro.

e. Copiar la ruta seleccionada en el descendiente, verificando que los clientes de la ruta seleccionada no se encuentren ya en el descendiente. Si ya existen eliminar este cliente de la ruta que se va a adicionar. Si todos los clientes se encuentran en el descendiente finalizar.

f. Al digito marcador de fin de ruta cambiarle el signo negativo por positivo.

g. Regresar al paso a.

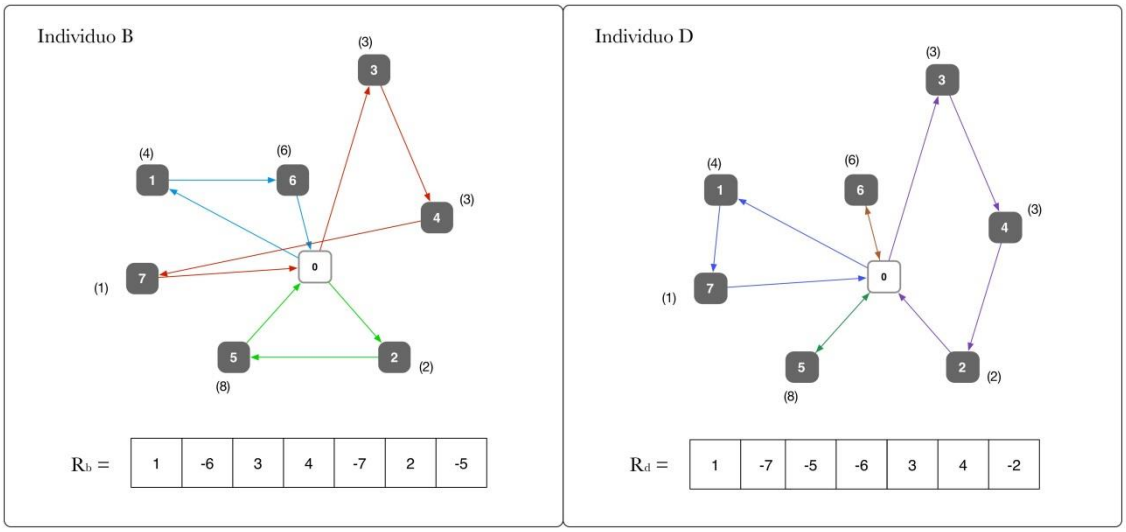
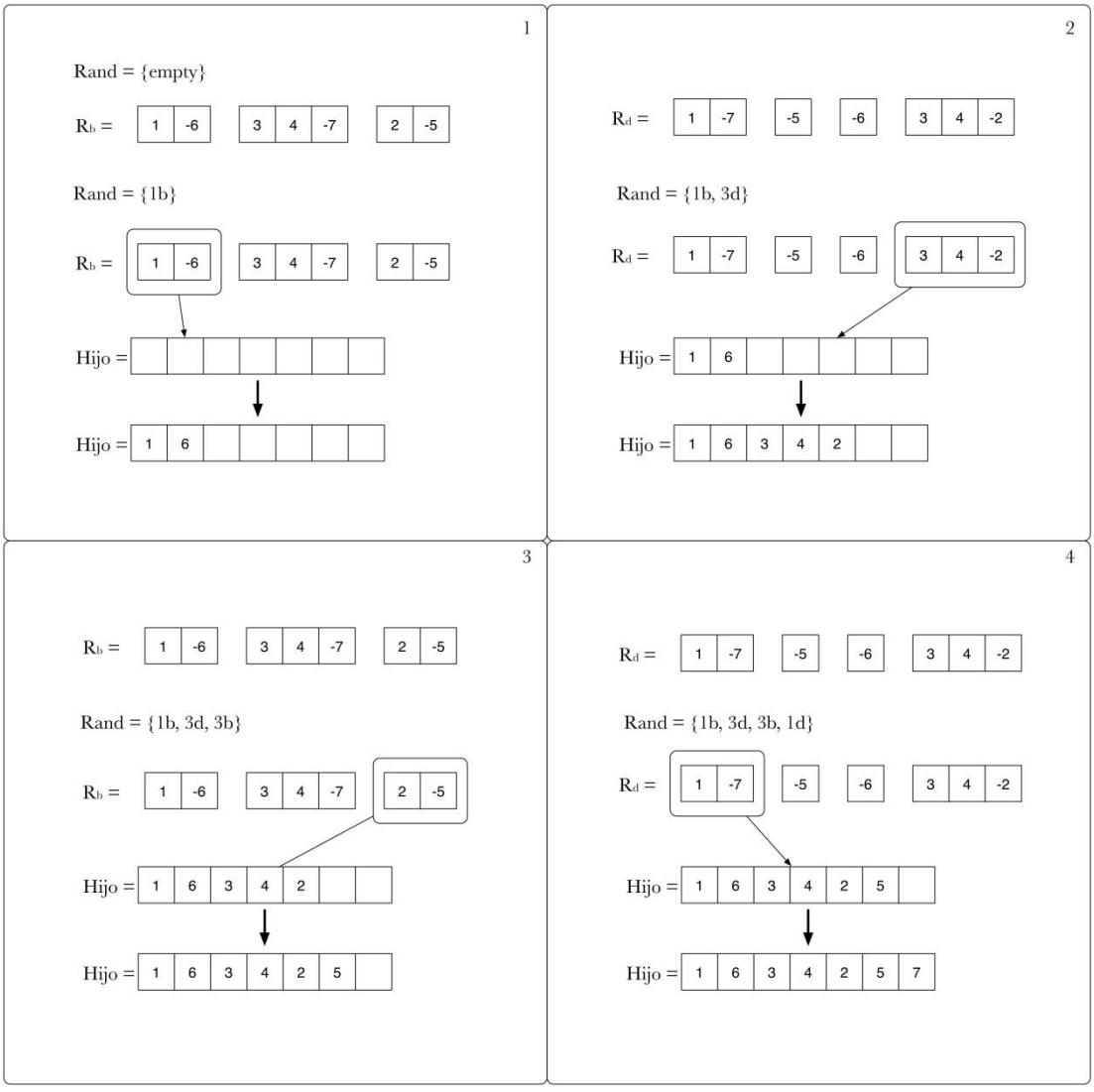
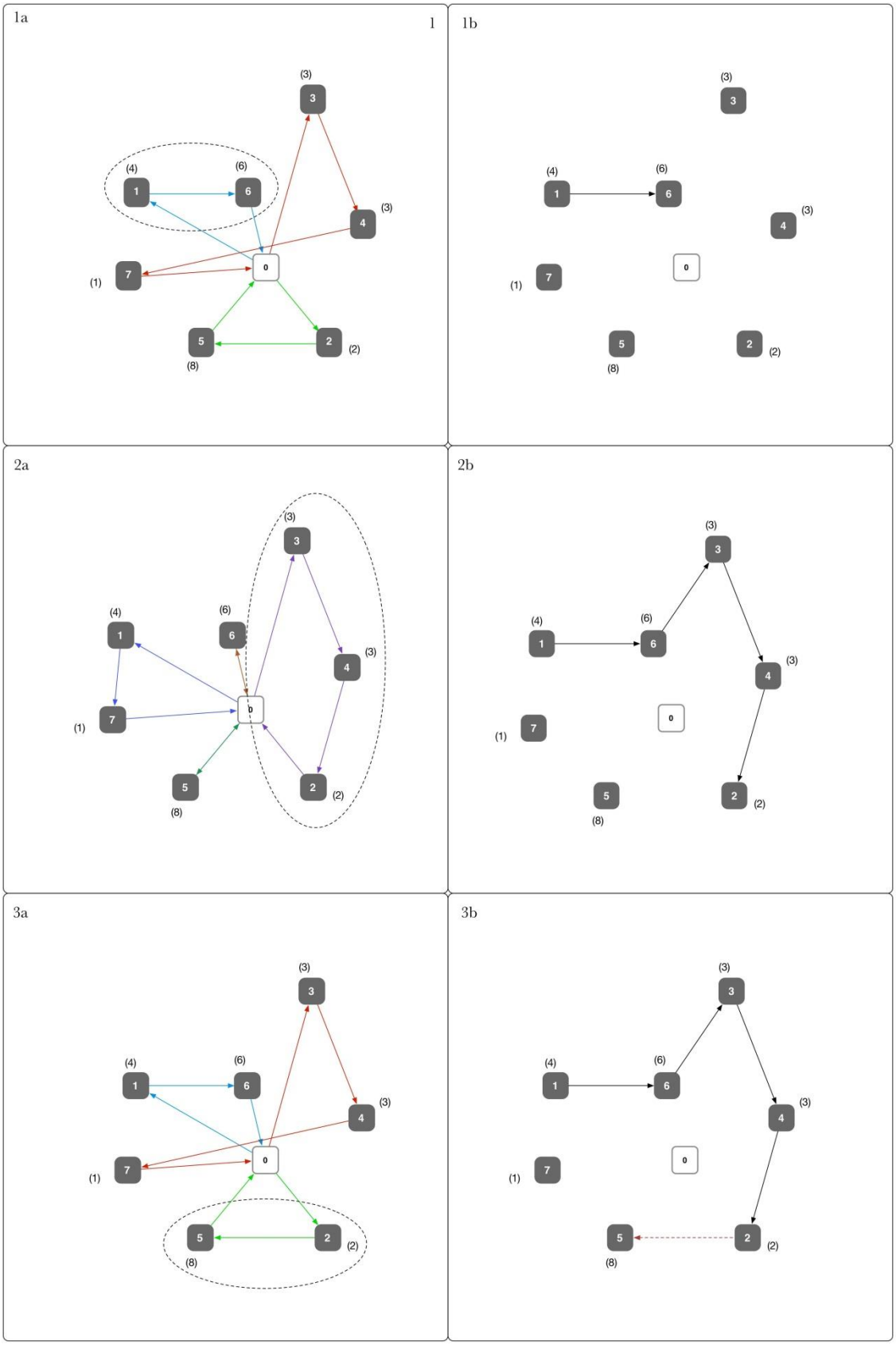


Figura 3.3. Individuos para recombinación




1a
1
1b
2a
2b
3a
3b

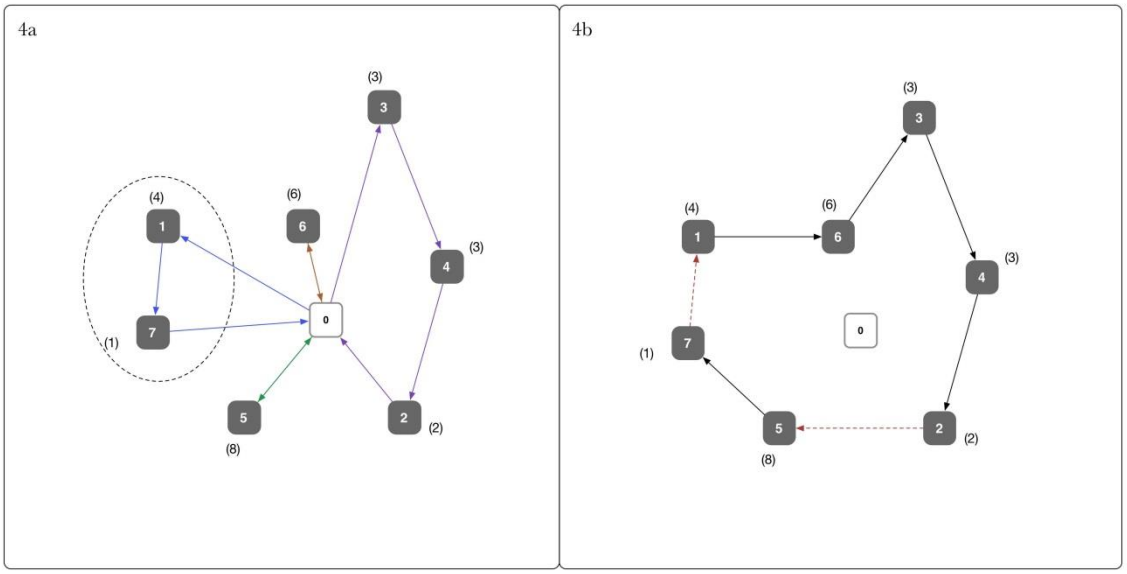


Figura 3.4. Recombinación por rutas representada en grafos.

3.7 Mutación

El descendiente obtenido en el paso anterior, el cual ya no contiene rutas, se somete a una cantidad variable de mutaciones simples, las cuales son normalmente conocidas como movimientos *swap* o *shift* [8]. Estos movimientos permiten realizar tanto desplazamientos de clientes intra-rutas e inter-rutas, ya que se realizan sobre el TSP resultante de la recombinación.

Posterior a la mutación del descendiente, empieza un procedimiento encargado en la división de la secuencia en rutas considerando la capacidad de los vehículos.

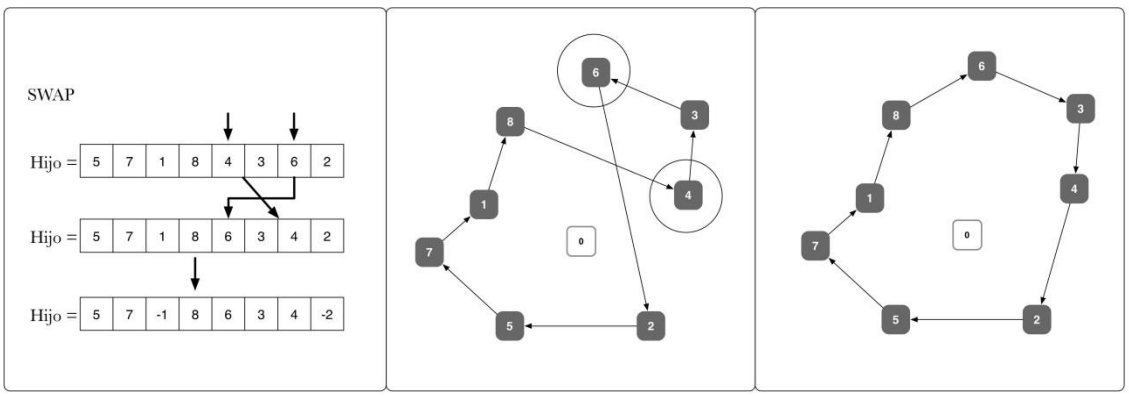


Figura 3.5 Mutación por movimiento *swap*

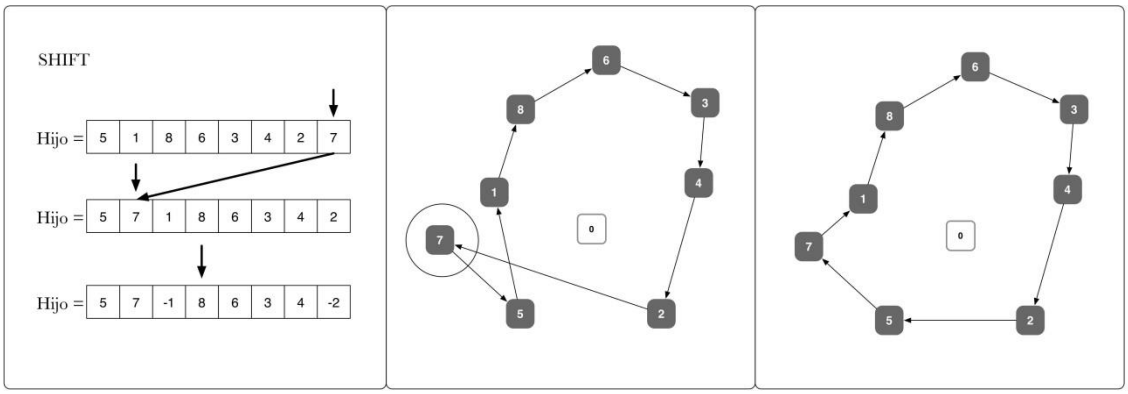


Figura 3.6 Mutación por movimiento *shift*

3.8 Etapa de mejoría propuesta: Metodología de enumeración implícita 0-1 o algoritmo de Balas

En el proceso de solución de cierta clase de problemas que usan la técnica de descomposición de Benders jerarquizada [19], se debe resolver de manera iterativa un problema de programación lineal entera (PLI). Para el problema específico de planeamiento de sistemas eléctricos de transmisión de energía eléctrica esto ocurre en la fase del proceso jerarquizado que corresponde al sub-problema de inversión, el cual es un problema de programación entera. Como se ha visto a lo largo del texto, existen diferentes métodos disponibles en la literatura para resolver problemas de PLI. En este capítulo se analiza el denominado algoritmo de enumeración implícita especializado basado en el algoritmo aditivo de enumeración implícita 0-1 de Balas. Las ventajas de este método, sobre otras técnicas que permiten resolver problemas de PLI, serán presentadas durante el desarrollo del capítulo.

El algoritmo desarrollado es adecuado para resolver problemas de PLI con variables limitadas. Por ejemplo, el problema de planeamiento de sistemas de transmisión de energía eléctrica a largo plazo [19], que es un problema real, posee esta característica, ya que las cantidades asociadas a los números de líneas y/o transformadores candidatos para ser adicionados en los distintos corredores del sistema deben ser representadas a través de variables enteras y limitadas.

Dentro de la formulación matemática de un problema que considera la posibilidad de que una variable entera, representada por x_k solo pueda asumir valores en un rango dado de valores, es decir, $x_k \leq u_k$, donde u_k es el valor máximo que puede asumir la variable, esta puede ser transformada en varias variables binarias a través de la siguiente relación:

$$x_k = \sum_{p=0}^{n} 2^p y_{kp} \; ; \; y_{kp} = 0, 1 \tag{3.1}$$

Donde n es determinado por la relación:

$$2^{n+1} \geq u_k + 1 \tag{3.2}$$

Esto quiere decir que el número entero x_k se representa a través de un número binario equivalente de $(n+1)$ bits, el cual se puede obtener a su vez, combinando las variables binarias elementales y_{kp}, que representan bits con diferente peso en el número binario. La ecuación (3.1) permite convertir a entero un número binario expresado a través de sus bits ordenados de menor peso: bit *0*, a mayor peso: bit *n*. Es importante tener en cuenta que para los fines del problema, las variables y_{kp} son variables binarias que solo pueden asumir el valor 0 ó 1, y que entre mayor sea el valor del índice p en esta variable, tiene más peso dentro de la respuesta en el caso de asumir el valor 1.

Para el caso en que $u_k = 3$ se tiene que *n* = 1, así, cada variable de inversión es transformada en dos variables binarias, las cuales quedan relacionadas de la siguiente forma:

$$x_k = y_{k0} + 2y_{k1}; \;\; y_{k0} = y_{k1} = (0, 1) \tag{3.3}$$

Para el caso $u_k = 3$ esto implica duplicar el número de variables enteras y limitadas al **"transformar"** nuestro problema en un problema de PLI (cero-uno), al cual se le pueda aplicar un algoritmo de enumeración implícita cero-uno.

La razón principal para utilizar algoritmos de enumeración implícita cero-uno especializados, los cuales generalmente se combinan con la técnica de descomposición de Benders jerarquizada para resolver problemas con variables enteras y continuas, es que, a diferencia de lo que ocurre con programación lineal (PL), no existe un algoritmo de programación entera que permita resolver todos los tipos de problemas, debido a que la eficiencia de un algoritmo de PI depende en gran medida de las características particulares del problema. Por lo tanto no existe un algoritmo de PI que sea mejor que los otros para resolver todos los problemas, ni programas de computación disponibles con tales características.

En este contexto, una de las primeras fases de un trabajo de investigación consiste en seleccionar un método para solucionar los problemas de PI

resultantes, y esta selección debe hacerse considerando los diferentes métodos que existen en la literatura especializada tales como: método de planos de corte, *Branch & Bound*, enumeración implícita, métodos subóptimos, etc. En una fase posterior se debe desarrollar un algoritmo especializado que aproveche las características específicas del problema.

En el caso ejemplo del problema de planeamiento de sistemas de transmisión, existe una metodología de solución basada en un algoritmo de enumeración implícita cero-uno especializado, que a su vez se construye a partir de las ideas del algoritmo aditivo de enumeración implícita cero-uno de Balas [2].

Las principales ventajas de un algoritmo de enumeración implícita son:

- Solamente realiza operaciones de adición y comparación, lo que elimina los errores de redondeo producidos en otros métodos, los cuales pueden generar inexactitudes en las respuestas o comprometer el desarrollo del método.

- No requiere de la solución de un PL subsidiario como es el caso de prácticamente todos los otros métodos. Sin embargo, existe un método de enumeración implícita con restricciones substitutas que requiere de la solución de un PL subsidiario.

- No cambia la matriz A y ocupa un espacio reducido de memoria.

- Permite gran flexibilidad en la implementación de las denominadas pruebas de sondaje.

Las características específicas de algunos problemas de la vida real permiten la implementación eficiente de un algoritmo de enumeración implícita cero-uno especializado.

3.8.1 Conceptos y definiciones de la Enumeración Implícita

La metodología presentada se desarrolla para un problema de PLI 0-1 que asume la siguiente forma general:

$$
\begin{aligned}
\min\ z = \ & \sum_{j \in N} c_j x_j\ ,\ c_j \geq\ 0 && j\ \in\ N = \{1, 2, ..., n\} \\
s.a. & & & \\
& \sum_{j \in N} a_{ij} x_j + S_i = b_i && i\ \in\ M = \{1, 2, ..., m\} \\
& x_j = (0, 1) && j\ \in\ N \\
& S_i \geq 0 && i\ \in\ M
\end{aligned}
\tag{3.4}
$$

Cualquier problema 0-1 puede ser escrito en la forma de la ecuación (3.4) observando que:

- Un problema de maximización es transformado en un problema de minimización multiplicando la función objetivo por -1.

- Cualquier $c_j < 0$ puede ser transformado a positivo redefiniendo la variable $x_j = 1 - x'_j$ y x'_j = (0,1).

- Todas las restricciones pueden ser escritas en la forma $\leq$ a las cuales después se les deben agregar las respectivas variables de holgura para asumir la forma de igualdad en las restricciones.

La idea de la enumeración implícita o parcial es intentar evaluar solamente una pequeña parte de todas las soluciones posibles, descartando las restantes por no ser interesantes. Una enumeración explícita de un problema de n variables requeriría la evaluación de 2^n soluciones posibles, lo que produce el fenómeno de explosión combinatorial cuando n es grande. A manera de ejemplo, consideremos el caso de un problema 1-0 con 100 variables binarias, las cuales pueden asumir arbitrariamente el valor 0 o el valor 1, y en el cual necesitamos evaluar todas las soluciones que resultan de combinar todas las posibilidades. Para este ejemplo, resultan 2^{100} posibilidades que es un número próximo a 10^{30} casos por evaluar. Suponiendo que tenemos una máquina de cálculo capaz de evaluar $4 * 10^{12}$ casos por segundo (máquina que aún no existe en la vida real), se evaluarían aproximadamente 10^{20} casos por año, por lo tanto se requerirían 10^{10} años para evaluarlas todas. Puesto que la edad aproximada del universo es de 10^{10} años, deberíamos haber usado aquella fantástica máquina desde que existe el universo para tener una respuesta en esta época. Este es apenas un caso de 100 variables binarias y algunos problemas de la vida real sobrepasan en mucho esta cantidad.

En la enumeración implícita se explora solamente una pequeña parte del espacio solución del problema y se encuentra la solución óptima del problema

en tiempos de cálculo razonables (segundos, minutos u horas). La implementación exitosa de esta metodología requiere de dos requisitos fundamentales:

1. El esquema de enumeración debe asegurar que todas las soluciones posibles sean enumeradas implícita o explícitamente y de una manera no redundante.

2. Se deben diseñar pruebas de sondaje que permitan excluir el mayor número posible de soluciones no interesantes o de baja calidad.

El esquema de enumeración de Glover [11] garantiza la primera condición. Respecto al segundo requisito, existen varias pruebas de sondaje propuestas, algunas de las cuales se presentan en este capítulo.

A continuación se presentan algunas definiciones y notaciones que serán utilizadas:

- **Notación:**

  $+j$ Indica que la variable $x_j = 1$

  $-j$ Indica que la variable $x_j = 0$

  Un elemento subrayado tal como $\underline{j}$ indica que la variable x_j para la alternativa $x_j = 0$ ya fue explorada y sondada.

- **Solución Parcial (J):**

  Es un conjunto ordenado donde se definen valores binarios a un subconjunto $j \subseteq N$. Por ejemplo:

  $$J = \{6, -2, -\underline{4}, 5\}$$

  Indica que $x_6 = x_5 = 1$, $x_2 = 0$ y $x_4 = 0$ con la alternativa $x_4 = 1$ ya está sondada, esto es, con esta alternativa ya explorada o eliminada por no ser interesante.

- **Variables Libres (N-J):**

  Son aquellas variables que aún no tienen un valor binario asignado por una solución parcial y, por lo tanto, se encuentra disponible para asumir un valor 0-1.

- **Complemento de J:**

  Es el conjunto de soluciones obtenidas a partir de J asignando a todas las variables que están aún libres valores binarios 0-1.

- **Solución Parcial Sondada:**

  Una solución parcial *J* puede ser sondada si todos sus complementos pueden ser descartados por no ser interesantes.

3.8.2 Esquema de Enumeración Implícita de Glover

El esquema de enumeración implícita de Glover [11] permite una implementación más adecuada del algoritmo de Balas.

En este esquema el proceso se inicia fijando el valor de una (o varias) variables y durante el proceso se obtiene una solución factible fijando el valor de nuevas variables. En el proceso muchos puntos de solución son excluidos implícitamente. La figura 3.1 muestra el esquema de enumeración de Glover.

Se puede mostrar que el esquema de enumeración de Glover mostrado en la figura 3.1 es finito y consigue enumerar implícita o explítamente, todas las 2^n posibles soluciones de una manera no redundante.

3.8.3 Algoritmo de Enumeración Implícita de Balas

En el algoritmo de Balas, durante la evolución del proceso, se almacena la mejor solución factible encontrada y después de enumerar todas las 2^n soluciones posibles implícita o explícitamente, la última mejor solución factible encontrada, denominada incumbente, es la solución óptima.

La figura 3.2 presenta el diagrama de flujo del algoritmo aditivo de Balas que resulta después de realizar algunas modificaciones sugeridas por Glover en su esquema de enumeración.

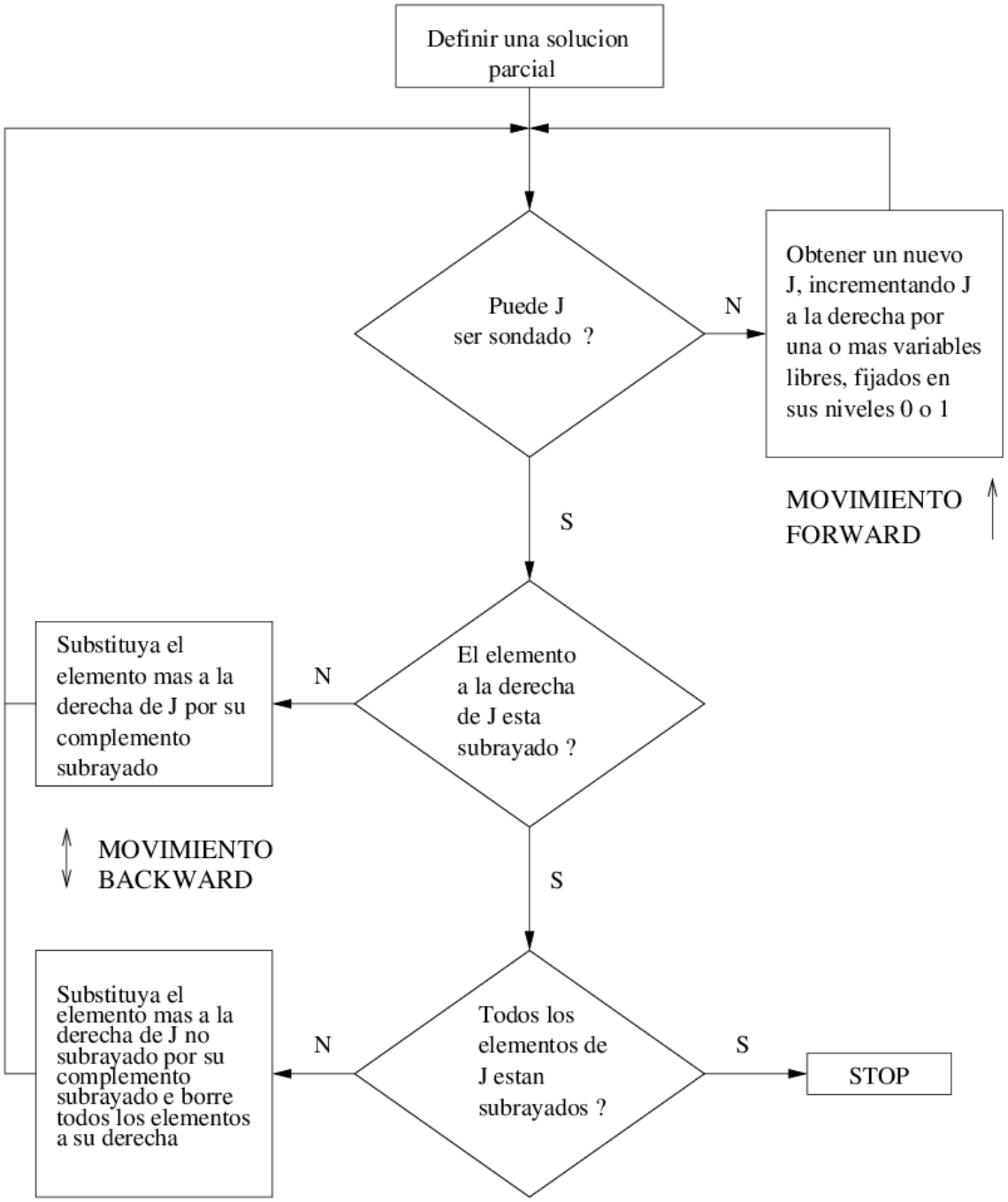


Figura 3.7. Esquema de Enumeración de Glover

El algoritmo general mostrado en la figura 3.2 puede asumir varios niveles de especialización después de definir las tareas específicas en cada uno de los bloques A,B,C,D y E. Por lo tanto, la especificación de las tareas en cada bloque define el nivel de especialización del algoritmo y en cierta forma, la potencia de la metodología. Se presentan las principales tareas desarrolladas en cada bloque con las particularidades incorporadas a nuestro algoritmo.

A. Prueba de Sondaje:

Las prueba de sondaje son diseñadas para excluir el máximo posible de complementos (esto es, soluciones derivadas) de una solución parcial por no ser interesantes. Estas pruebas son fundamentalmente de tipo heurístico y pueden ser tan débiles que permitan la enumeración explícita de casi todas las 2^n soluciones factibles o tan poderosas que excluyan prácticamente todas las soluciones posibles no interesantes.

En la iteración t, sea J_t, la solución parcial, entonces se tiene:

$$S_i^t = b_i - \sum_{j \in J_t; j>0} a_{ij} \; ; \; i \; \in \; M \tag{3.5}$$

$$z^t = \sum_{j \in J_t; j>0} c_j \tag{3.6}$$

Donde S_i^t define el valor de las variables de holgura y z^t la función objetivo. Sea z_{min}la mejor solución factible encontrada, denominada incumbente. La idea de las pruebas de sondaje, en la tentativa de excluir un conjunto de posibles soluciones porque son consideradas no interesantes, puede ser hecha en base a dos consideraciones básicas mostradas a continuación. Para la solución parcial definida por J_t, suponer que S_i^t < 0 para por lo menos un $i \in M$. En esta situación se puede deducir que:

- J_t no tiene complemento factible.
- J_t tiene complemento factible pero con una función objetivo mayor que la incumbente, esto es, el mejor objetivo z_c^t que se puede obtener a partir de J_t lleva a un $z_c^t > z_{min}$.

En ambos casos J_t es sondado y debe ser hecho un movimiento backward. Si las pruebas de sondaje fallan entonces se debe hacer un movimiento forward intentando inicialmente asignar o especificar valores definidos a algunas variables libres a través de las denominadas pruebas de ampliación o entonces asignar el valor 1 a una variable libre, definido en el bloque C en la figura 3.2, en la tentativa de encontrar un mejor complemento factible.

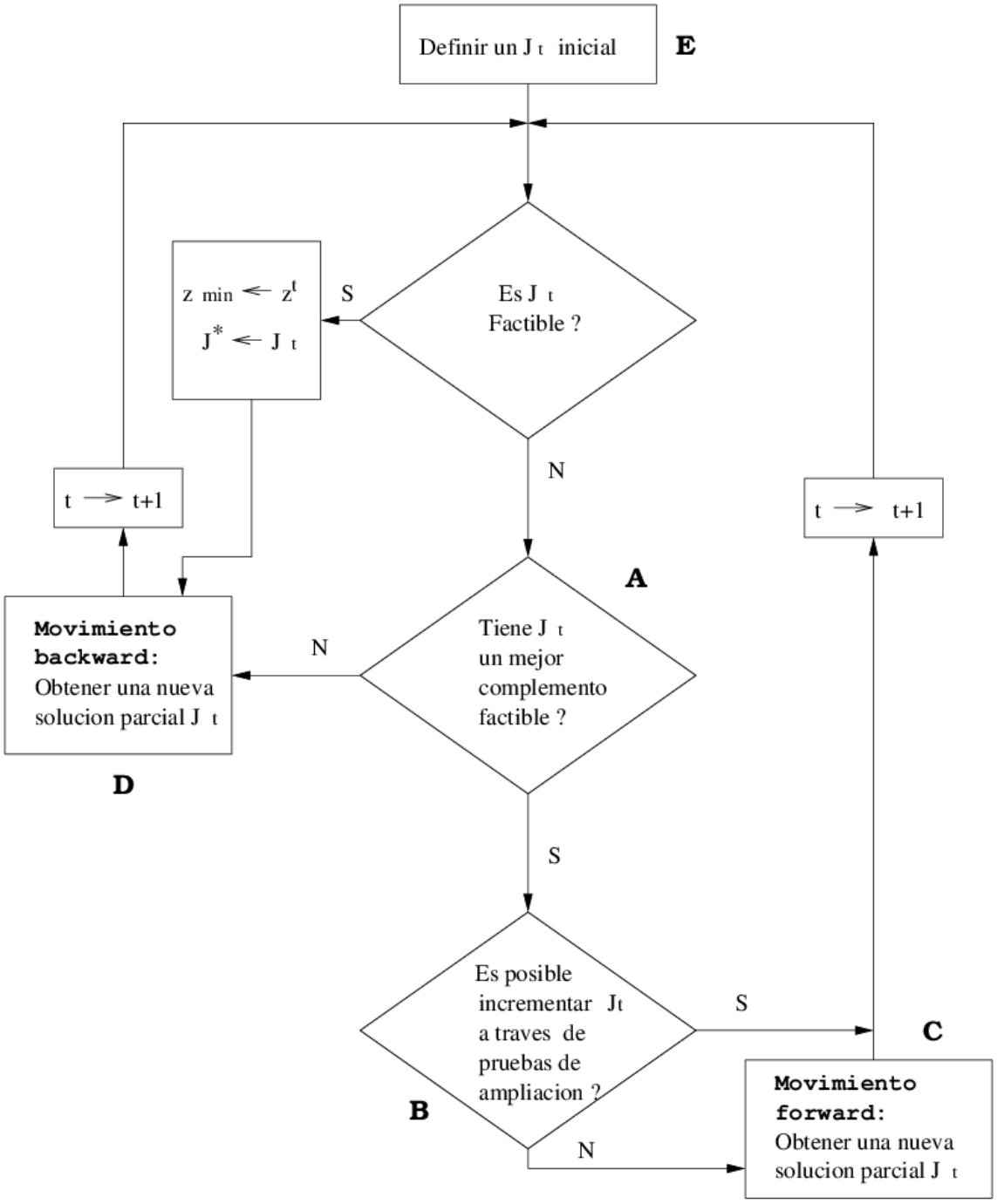


Figura 3.8. Algoritmo Aditivo de Balas

Se presentan las pruebas de sondaje de Balas y Glover-Zionts [10] que son implementadas en el algoritmo. Para mostrar y/o demostrar estas pruebas de sondaje es conveniente representar las expresiones generales de S_i^t y z^t en la siguiente forma:

$$S_i^t = b_i - \sum_{j \in J_t, j>0} a_{ij} - \sum_{j \in J_t, j<0} a_{ij} - \sum_{j \in (N-J_t)} a_{ij} \tag{3.7}$$

$$z^t = \sum_{j \in J_t\ j>0} c_j + \sum_{j \in J_t\ j<0} c_j + \sum_{j \in (N-J_t)} c_j \tag{3.8}$$

Prueba 1: (Balas)

Se define:

$$A_t = \{j \ \in \ (N - J_t) \mid a_{ij} \ \geq \ 0, \ \ \forall i \ \mid \ S_i^t < 0\} \tag{3.9}$$

Los elementos de A_t son aquellas variables libres que al ser elevadas al nivel 1 no mejoran la infactibilidad de la solución parcial actual. Esto puede ser visto en (3.7) pues si $x_j;\ j \ \in \ A_t$ es elevado al nivel 1 entonces la infactibilidad de S_i^t aumenta. Así todos aquellos $x_j;\ j \ \in \ A_t$ son excluídas pues no contribuyen en el proceso de eliminar la infactibilidad.

Sea: $N_t^1 = (N - \ J_t) - \ A_t$

Si : $N_t^1 \neq \ \emptyset$, esto significa que ninguna variable libre puede ser elevada al nivel 1, entonces J_t es sondado por infactibilidad y, se debe hacer un movimiento *backward*.

Prueba 2: (Balas)

Es una prueba de exclusión de variables, del vector de variables libres, que usa un criterio de optimización, en donde se define:

$$B_t = \{j \ \in \ N_t^1 \ \mid \ (z^t \ + c_j) \ \geq \ z_{min}\} \tag{3.10}$$

Los elementos de B_t son aquellas variables libres que a pesar de que puedan mejorar la infactibilidad del problema, cada variable lleva a una función objetivo

mayor que la incumbente z_{min}, esto es a un objetivo de peor calidad. Así, los $x_j;\ j\ \in\ B_t$ son excluídos como candidatos para asumir el valor 1 pues no son interesantes por optimalidad.

Sea: $N_t^2 = N_t^1 - B_t$

Si $N_t^2 \neq \emptyset$, esto significa que ninguna variable libre puede ser elevada al nivel 1, entonces J_t es sondado pues no existe un complemento factible mejor que la incumbente actual. Así, J_t **es sondado** porque no tiene mejor complemento factible y, se debe hacer un movimiento *backward.*

Prueba 3: (Balas)

Es una prueba de sondaje, en donde se define:

$$C_t = \left\{ i\ \in\ M\ \mid\ S_i^t < 0\ ;\ \sum_{j \in N_t^2} a_{ij}^- > S_i^t \right\} \qquad (3.11)$$

donde:

$$a_{ij}^- = \min(0, a_{ij})$$

Los elementos de C_t son los índices de aquellas restricciones infactibles en donde a pesar de elevar todos los elementos de N_t^2 a nivel 1, la restricción continua siendo infactible. En la ecuación (3.11) se puede ver claramente que si una restricción tiene su holgura actual en un valor negativo y usamos todas las variables libres aún disponibles con $a_{ij} < 0$, con el propósito de mejorar la infactibilidad, esta restricción aún continua siendo infactible, entonces no existe complemento factible de J_t. Si $C_t \neq \emptyset$, por lo menos una restricción continuará siendo infactible, entonces J_t es sondado, pues no tiene complemento factible. Se debe entonces realizer un movimiento *backward*. Si $C_t = \emptyset$, se intenta aún sondar J_t a través de la prueba 3'.

Prueba 3': (Glover-Zionts)

En esta prueba se evalúa si cada variable $x_j \in N_t^2$ al ser promovida a su valor 1 producirá un aumento en la función objetivo más allá de la relación permisible para la restricción violada.

Para cada $S_i^t < 0$ calcular:

$$r_i = \min_{j \in N_t^2} \left\{ \left(\frac{S_i^t}{a_{ij}^-}\right) c_j; \quad a_{ij}^- < 0 \right\} \tag{3.12}$$

donde $\left(\frac{S_i^t}{a_{ij}^-}\right) c_j$ es el costo relativo, de la variable x_j, respecto al grado de reducción de la infactibilidad de dicha variable en la restricción violada i.

Si $r_i \geq (z_{min} - z^t)$ entonces J_t es sondado.

Esta prueba diseñada por Glover-Zionts [10], fue motivada por el hecho de que la prueba 3 de Balas podría ser ineficiente en algunos casos, por ejemplo en muchos casos se puede satisfacer la factibilidad pero llevando el valor del objetivo a un valor mayor que la incumbente lo que significa que la solución parcial J_t realmente podría ser sondada sin embargo ésta posibilidad no es detectada por la prueba 3 de Balas. En otras palabras, puede ocurrir que:

$$\sum_{j \in N_t^2} a_{ij}^- < S_i^t \tag{3.13}$$

Al promover aquellas variables al nivel 1 llevaría al valor del objetivo parcial a un valor mayor que la incumbente.

El siguiente ejemplo ilustra este hecho: Suponer que en un determinado paso en la solución de un problema se tiene $z_{min} = 40$; $N_t^2 = 3{,}5$; $z^t = 18$ y algunos datos parciales mostrados en la siguiente tabla:

Variable	x_3	x_5	
Restricción	$a_{23} = -6$	$a_{25} = -4$	$S_2 = -8$
Costo	$c_3 = 12$	$c_5 = 20$	

Esta solución parcial no es sondado por ninguna de las pruebas de sondaje de Balas (1,2,3). Especialmente en la prueba 3 de Balas ocurre que -6 -4 = -10 < -8 lo que llevaría a $S_2 = 2$, esto es, haciendo $x_3 = x_5 = 1$ se elimina la infactibilidad de la restricción 2 y la prueba de sondaje 3 de Balas falla pues no consigue sondar esta solución parcial. Sin embargo, puede observarse que eliminar esta infactibilidad llevaría la solución parcial a $z^t = 18 + 12 + 20 = 50$ lo que es peor que la incumbente: $z_{min} = 40$. La prueba de Glover-Zionts fue diseñada para lograr el sondaje en este tipo de casos. A continuación se resume la prueba de Glover-Zionts.

Prueba de Glover-Zionts:

Una condición suficiente para sondar una solución parcial J_t es la siguiente:

Si existe un $i \mid S_i^t < 0$ y además:

$$\frac{S_i^t c_j}{a_{ij}^-} \geq (z_{min} - z^t) \;\; \forall j \in N_t^2; \quad a_{ij}^- < 0 \tag{3.14}$$

Entonces no existe complemento factible de J_t que puedan mejorar la incumbente, entonces la solución parcial definida por J_t es sondada.

Prueba: Prueba por lo absurdo.

Asumir que existe una solución mejor que la incumbente. Entonces para tal solución tenemos que $\sum a_{ij}^- \leq S_i^t$, que es el caso que realmente interesa, donde la suma es hecha solamente con los elementos negativos de $a_{ij} \mid j \in N_2^t$ y a_j está en la base para la solución especificada. La relación anterior, que es la misma presentada en (3.13), puede ser escrita en la siguiente forma:

$$\sum a_{ij}^{-} = \rho S_i^t; \quad \rho > 1 \tag{3.15}$$

Entonces si $\sum c_j$ está restricto a esos mismos valores de j, el valor de la función objetivo actual definido por $z^t + \sum c_j$ llevaría a un límite inferior de la incumbente pues estamos suponiendo que existe una mejor incumbente, entonces tenemos:

$$z^t + \sum c_j < z_{min}$$

Por otro lado, por la hipótesis, tenemos que:

$$\begin{aligned}
z^t + \sum c_j &= z^t + \sum a_{ij}^{-} \left(\frac{c_j}{a_{ij}^{-}}\right) \\
&\geq z^t + \sum a_{ij}^{-} \left[\frac{z_{min} - z^t}{S_i^t}\right] \qquad \text{de (3.14)} \\
&\geq z^t + \left(\frac{\sum a_{ij}^{-}}{S_i^t}\right)(z_{min} - z^t) \\
&\geq z^t + \rho(z_{min} - z^t) \qquad \text{de (3.15)} \\
&\geq \rho z_{min} - (\rho - 1) z^t \\
&\geq \rho z_{min} - z_{min} + z_{min} - (\rho - 1) z^t \\
&\geq z_{min} + (\rho - 1)(z_{min} - z^t) \geq z_{min} \\
z^t + \sum c_j &\geq z_{min}
\end{aligned}$$

y en consecuencia se llega a una contradicción, lo que permite demostrar la validez de la prueba de Glover-Zionts por lo absurdo.

Por lo tanto, de acuerdo con Glover-Zionts se puede resumir que:

Para aquellas restricciones $i \mid S_i^t < 0$ verificar la relación:

$$\frac{S_i^t c_j}{a_{ij}^{-}} \geq (z_{min} - z^t) \qquad \forall j \in N_t^2$$

Si la relación es verdadera para cualquier i entonces $J_t$es sondado.

B. Movimiento Forward: Prueba de ampliación de variables

Cuando las pruebas de sondaje fallan entonces se debe incrementar el número de variables en J_t, esto es, algunas variables libres deben asumir valores definidos y deben ser incluidas en el conjunto J_t asumiendo valores específicos 0 o 1. Un primer intento de incremento de variables consiste en realizar las denominadas pruebas de ampliación, las cuales permiten determinar el valor exacto 0 o 1 de alguna variable libre para garantizar condiciones de factibilidad a la solución parcial definida por J_t. En este capítulo se presentan y se usan dos pruebas de ampliación de variables: las pruebas de ampliación de variables de Glover-Zionts y de Geoffrion [9].

Prueba de Ampliación de Geoffrion:

Teorema:

1) La restricción:

$$\beta - \sum_j \alpha_j x_j \geq 0 \ \ (>0), \ \ x_j = (0,1)$$

Es infactible binaria, esto es, no tiene solución binaria factible si y solamente si:

$$max \left\{ \beta - \sum_j \alpha_j \, x_j \ \mid \ x_j \ = \ (0,1) \right\} \ = \ \beta \ - \ \sum_j \ min \ (0, \ \alpha_j) < 0 \quad (\leq \ 0)$$

2) En cualquier solución binaria:

$$\beta - \sum_j \ \alpha_j x_j \ \geq 0 \quad (>0)$$

$$\beta - \sum_j \ min \ (0, \ \alpha_j) - |\alpha_{jo}| < \ 0 \implies x_{jo} = 0 \ \text{ o } \ x_{jo} = 1$$

Si $\propto_{j0} > 0$ o $\propto_{j0} < 0$ respectivamente.

Prueba: La prueba es simple. Se prueba solamente la primera parte de 1.

1) Si:

$$\beta - \sum_j \alpha_j x_j \geq 0$$

Es binario infactible entonces:

$$\max \left[\beta - \sum_j \alpha_j x_j \right] < 0$$

$$\max \left[\beta + \sum_{j;\alpha_j<0} -\alpha_j x_j + \sum_{j;\alpha_j\geq 0} -\alpha_j x_j \right] < 0$$

$$\beta + \max \sum_{j;\alpha_j<0} -\alpha_j x_j + \max \sum_{j;\alpha_j\geq 0} -\alpha_j x_j < 0$$

$$\beta + \sum_{j;\alpha_j<0} -\alpha_j + \sum_{j;\alpha_j\geq 0} 0 < 0$$

$$\beta - \sum_j \min(0, \alpha_j) < 0$$

La utilización de la parte 2 del teorema de Geoffrion lleva a una forma simple de fijar los valores de algunas variables libres. Así, la prueba puede ser establecida de la siguiente forma:

Prueba de Ampliación de Geoffrion: Para cada i tal que $S_i^t < 0$ y cada $j \in N_t^2$, si:

$$\left[S_i^t - \sum_{j\in(N_t^2)} \min(0, a_{ij}) - |a_{ij}| \right] < 0 \qquad (3.16)$$

Entonces

$$x_j = 0 \text{ si } a_{ij} > 0$$

O

$$x_j = 1 \text{ si } a_{ij} < 0$$

Prueba de Ampliación de Glover-Zionts:

Esta prueba permite fijar una variable libre en el nivel 0 a través de una prueba muy simple formulada de la siguiente forma:

Para cada i y cada $p \in N_t^2$ tal que $a_{ip} > S_i^t$, calcular:

$$c_h = \min \{c_j \mid j \in N_t^2 - \{p\}, \ a_{ij} < 0\} \tag{3.17}$$

Si:

$$c_h + c_p \geq z_{min} - z^t$$

Entonces:

$$x_p = 0$$

C. Movimiento Forward: Criterio de Incremento de Variables

Si las pruebas de sondaje y de ampliación de variables fallan se debe seleccionar una variable libre j para ser adicionada a J_t con un valor de $x_j = 1$. Esta variable es seleccionada a través de la denominada prueba 4 de Balas.

Prueba 4: (Balas)

Seleccione una variable x_{j*} , $x_{j*} \in N_t^2$ para asumir el nivel 1 y adicionar *j** a J_t. Esta variable es seleccionada a través de la relación:

$$v_{j^*} = \max_{j \in N_t^2} \{v_j\} \tag{3.18}$$

Donde

$$v_j = \sum_{i \in M} \min(0, S_i^t - a_{ij});\ j \in N_t^2 \tag{3.19}$$

v_j es una medida empírica de la infactibilidad total de la nueva solución parcial después de hacer $x_j = 1$. En caso de empate j^* es seleccionado de tal manera que c_{j*} sea el menor entre los c_j candidatos.

Si $v_{j*} = 0$ entonces no existe infactibilidad y así J_{t+1} es factible y esta nueva solución debe llevar a un mejor valor de la incumbente z_{min}. Entonces J_{t+1} es sondado y se debe actualizar la incumbente.

Este criterio de entrada de variables que cuantifica la infactibilidad total de la solución parcial después de hacer $x_j = 1$ no es el único criterio de selección de variables para adicionar a J_t. Para algunos problemas, otros criterios pueden ser más eficientes tales como:

- Seleccione la variable x_j que produce la menor infactibilidad en la restricción más violada.
- Seleccione la variable x_j que elimine la infactibilidad del mayor número de restricciones.
- Seleccione una variable x_j que tiene un c_j pequeño y una gran capacidad de reducir la infactibilidad total de la solución parcial con la intención de encontrar buenas soluciones factibles.

D. Movimiento Backward

Si una solución parcial ha sido sondada entonces se debe hacer un movimiento *backward* en el esquema de Balas, es decir, debe modificarse el valor de una variable de J_t. En el esquema normal de Balas este proceso se realiza a través de la regla LIFO para ordenar las variables. En la regla LIFO (*last-input, first-out*) la última variable en entrar en la lista es la primera considerada para futuras exploraciones. Por ejemplo, sea J_t.

$$J_t = \{2, -\underline{5}, 3, 6, \underline{4}\} \tag{3.20}$$

Si J_t es sondado, la nueva solución parcial es definida por la relación:

$$J_{t+1} = \{2, -\underline{5}, 3, 6, -\underline{1}\} \tag{3.21}$$

Evidentemente, el orden en que se analizan las variables altera el proceso de enumeración. Este hecho fue observado por Tuan que demostró que no es necesario seguir estrictamente la regla LIFO, implícitamente embebida en el esquema de enumeración de Glover, para garantizar un desarrollo correcto del esquema de enumeración de Glover. Tuan sugiere la determinación de un subconjunto J'' constituido por aquellas variables que pueden ser seleccionadas para el desarrollo de futuras exploraciones, esto es, de aquellas variables que pueden ser subrayadas. Si j_1 es el elemento de J_t que sería seleccionado por la regla LIFO, entonces los elementos de J'' son todos aquellos elementos de J_t que están localizados a partir de j_1 inclusive hasta encontrar el primer elemento subrayado en un tramo de derecha a izquierda en los elementos de J_t. Así, por ejemplo, si J_t es definido por (3.20), entonces:

$$J'' = \{3, 6, 1\}$$

Por lo tanto, en un movimiento *backward*, de acuerdo con la idea de Tuan, cualquier elemento de J'' puede ser complementado o subrayado. En el ejemplo, a partir de J_t se pueden generar las siguientes soluciones parciales:

$$J_t = \{2, -\underline{5}, 3, 6, -\underline{1}\}$$

$$J_t = \{2, -\underline{5}, 3, 1, -\underline{6}\}$$

$$J_t = \{2, -\underline{5}, 6, 1, -\underline{3}\}$$

Según Tuan [23], el orden de los elementos 3, 6 y 1 es indiferente y, lo que realmente es importante es el orden y la posición de los elementos subrayados. Obviamente cada una de las 3 alternativas mostradas anteriormente llevan a estructuras de árbol y convergencia diferentes. Así, siempre existe la posibilidad de seleccionar aquella variable que puede entregar las mejores condiciones de sondaje.

El criterio para seleccionar un elemento de J'' es permitir condiciones favorables para producir complementos factibles rápidamente.

Prueba de Tuan:

El elemento $j \in J_t''$ a ser complementado, esto es igualado a 0 y subrayado, es aquel que produce en la solución parcial resultante la menor cantidad de infactibilidad total. Así, el elemento p seleccionado debe satisfacer:

$$w_p = \max_{j \in J_t''} \{w_j\} \tag{3.22}$$

Donde:

$$w_j = \sum_{i \in M} \min(0, S_i^t + a_{ij}); \quad j \in J'' \tag{3.23}$$

Donde w_j cuantifica la infactibilidad cuando $x_j = 0$. En el caso de empate, p define aquella variable que tiene el mayor c_j entre las variables que empataron.

E. Determinación de una Solución Inicial

Al comenzar el proceso se puede usar una solución inicial factible, la cual se convierte en la incumbente inicial o, se puede iniciar con una incumbente:

$$z_{min} \Rightarrow \infty \text{ y } J_t = \emptyset.$$

En el algoritmo normal de Balas se inicia el proceso con todas las variables libres y $z_{min} = \infty$. El proceso rápidamente entrega una solución factible pero de baja calidad. Se ha observado también que un conocimiento a priori de una solución factible de buena calidad y su uso como incumbente inicial reduce considerablemente el proceso de enumeración implícita.

Una buena solución inicial que puede ser sub-óptima puede ser encontrada a través de métodos heurísticos utilizando tiempos computacionales reducidos. Existen varios algoritmos heurísticos que pueden ser implementados, algunos simples y otros más sofisticados.

Otra alternativa para determinar una *buena* solución inicial es sugerida por Petersen [2]. En esta alternativa se sugiere hacer un ordenamiento de las variables seleccionando primero aquellas variables que tienen los mejores valores de la relación:

$$\frac{\frac{1}{c_j}}{\sum a_{ij}}$$

Una variante más simple es hacer un ordenamiento considerando solamente la restricción más restrictiva, esto es considerando solamente aquella restricción que tiene:

$$\max\left\{\sum a_{ij} - b\right\}$$

Sin embargo se ha observado que la eficiencia de esta alternativa depende de la estructura del problema.

Se puede además usar el conocimiento a priori del valor de algunas variables en la solución óptima. Esta idea, sugerida por Geoffrion [9], consiste en determinar un J_t inicial constituido por una combinación de estas variables. Se puede también ordenar estas variables en J_t según el *grado de certeza* de sus valores en la solución óptima. Así, por ejemplo, si tenemos casi la certeza total de que $x_7 = 1$ en la solución óptima, mucha certeza de que $x_2 = 0$ y poca certeza de que $x_5 = 1$, entonces $J_t = \{7, -2, 5\}$ es mejor que $J_t = \{5, -2, 7\}$ o cualquier otra combinación posible. La misma estrategia puede utilizarse para seleccionar una permutación cuando J_t es sondada lo que llevaría a una variante de la prueba de Tuan.

3.8.4 Algoritmo de Enumeración Implícita

Además de las características anteriormente mencionadas, que son propias de un algoritmo de enumeración implícita de carácter general, se deben también utilizar, en un algoritmo especializado, las características propias del problema a resolver, especialmente aquellas que tienen relación con la estructura del problema, conocimiento previo de las particularidades del problema y un especial conocimiento de las características de las variables físicas las cuales representan las variables binarias del problema. Así, un conocimiento de las características particulares de un problema puede llevar al diseño de un algoritmo de enumeración implícita especializado que sea muy eficiente al aprovechar esas informaciones disponibles en la aceleración del proceso de solución.

En relación a un problema cuyas variables de decisión pueden asumir valores enteros, por ejemplo: 0,1,2,3, etc., estas variables deben ser transformadas en variables binarias para ser implementadas en un algoritmo de enumeración implícita 0-1. En el siguiente análisis se ha limitado el valor máximo que puede asumir una variable a 3.

En consecuencia, se tiene un número de variables binarias igual al doble de las variables de inversión lo que, en principio, representa un aumento significativo en el tamaño del problema.

Por otro lado, el algoritmo de enumeración implícita puede ser usado en la fase 3 del proceso de solución en problemas donde se utiliza descomposición de Benders jerarquizada. En esta fase se resuelve un subproblema con variables enteras. Esta solución iterativa del subproblema permite disponer de mucha información de las características del problema pues, en el proceso iterativo un

subproblema en la iteración *(k + 1)* es simplemente el mismo problema en la iteración *k* con una restricción adicionada: el último corte de Benders generado [19]. De esta forma es posible aprovechar mucha información disponible del proceso de solución del problema en la iteración *k* con el propósito de resolver más rápidamente el problema en la iteración *(k + 1).*

Es importante, de igual manera, observar que la evolución del proceso iterativo es gradual, esto es, el incremento de la función objetivo de una iteración a otra es relativamente pequeño lo que, en otras palabras, hace un acontecimiento casi imposible por ejemplo que una variable entera con $x_j = 0$ en las soluciones óptimas del problema en las *k* primeras iteraciones pase a $x_j = 2$ en la iteración *(k + 1).*

Al analizar el problema de planeamiento de la expansión de sistemas de transmisión, en el cual ha sido usado este algoritmo con éxito, se observa que en el proceso de solución del subproblema de inversión se tienen las siguientes características que pueden ser aprovechadas en la especialización de un algoritmo de enumeración implícita 0-1:

1) Se puede reducir significativamente el número de variables binarias debido a la evolución gradual de los costos de inversión en el proceso iterativo. En consecuencia, solamente son duplicadas aquellas variables que ya alcanzaron valores diferentes de cero en los procesos iterativos previos. Se ha observado que un número reducido de las variables asumen valores diferentes de cero en la solución óptima de los subproblemas de inversión lo que lleva a duplicar un número reducido de las variables de inversión en variables binarias. De esta forma se reduce considerablemente el número de variables binarias.

2) Los elementos de la matriz A, obtenidos a partir de los cortes de Benders, son prácticamente todos ellos no negativos y un número muy reducido de ellos son significativos, por lo menos aquellos obtenidos de los cortes de Benders en la fase 1 del proceso iterativo lo que convierte estas restricciones en especialmente adecuadas para las pruebas de sondaje en el propósito de eliminar el mayor número de complementos de una solución parcial por ser infactibles o no interesantes.

3) El número de iteraciones en un algoritmo de enumeración implícita depende en gran medida de las características del problema. Se ha observado también, cuando se utiliza el método de Balas, que en problemas donde en la solución óptima existe un número reducido de variables con valor 1, las pruebas de sondaje son relativamente

eficientes produciendo una rápida convergencia del algoritmo. La experiencia refuerza la observación de Balas y también muestra que este hecho es una característica muy particular de algunos problemas de la vida real, y el consecuente subproblema de variables enteras, donde se tiene un elevado número variables enteras, pero en la solución óptima solamente un número muy reducido de estas asumen valores diferentes a cero, tradicionalmente entre 10 a 20% de las variables, lo que hace de este tipo de problema **naturalmente adecuado** para ser resuelto por un algoritmo de enumeración implícita.

4) Existe mucha información disponible del proceso de solución del sub-problema de inversión en la iteración *k* que puede ser adecuadamente aprovechada en la iteración *(k + 1)*. La diferencia entre los dos subproblemas es solamente una restricción adicional (último corte de Benders generado). Así, se puede tener la certeza casi completa de que la mayoría de las variables que tenían valores diferentes de cero en la iteración *k*, también asumirán valores diferentes de cero en la iteración (*k+1)* pues el cambio entre una iteración y otra es muy pequeño.

   Por tanto, aquellas variables que asumieron valores diferentes de cero en la solución óptima del problema en la iteración *k* pueden ser utilizadas, ordenadas adecuadamente como sugiere Geoffrion, para obtener una buena solución inicial o incumbente del problema en la iteración *(k + 1),* pues aquellas variables pueden ser incluídas prioritariamente en la obtención del J_t inicial, o sea, una buena solución inicial factible.

   Otra forma de producir una buena solución inicial es almacenar un subconjunto constituído por parte de las incumbentes generadas en la solución del problema en la iteración *k* y en las iteraciones anteriores, pues varias de ellas también pueden ser factibles para el problema en la iteración *(k + 1)* convirtiéndose de esta manera, la mejor de ellas, en excelente incumbente inicial para este nuevo problema.

   Este proceso de almacenamiento de incumbentes puede ser mantenido y actualizado en el transcurso de todo el proceso iterativo, eliminando aquellas incumbentes que se volvieron infactibles para los nuevos problemas y, eventualmente almacenando nuevas incumbentes que sean generadas en el proceso de solución. Además de eso, este proceso es favorecido de manera especial en este tipo de algoritmo pues el proceso de almacenamiento de incumbentes no implica computación adicional y ocupa espacio reducido de memoria. Esta idea es reforzada por el hecho de que el requerimiento de memoria es un

problema inexistente en los algoritmos de enumeración implícita donde la ocupación de memoria es insignificante comparado con los otros métodos.

5) También se ha observado en el método de Balas, que el incremento del número de restricciones generalmente aumenta la eficiencia de los criterios o las pruebas de sondaje reduciendo, en ocasiones de manera considerable, el número de iteraciones. En el caso del problema de planeamiento de sistemas de transmisión además de los cortes de Benders generados, que son las restricciones en el subproblema de inversión, se pueden también generar otros cortes o restricciones adicionales en la tentativa de aumentar la eficiencia de las pruebas de sondaje. Existen varios tipos de restricciones que pueden ser adicionadas al subproblema de inversión.

3.8.5 Algoritmo de Enumeración Implícita Cero-Uno:

Basado en los tópicos presentados anteriormente se elabora el siguiente algoritmo de enumeración implícita especializado, donde *k* es el contador de iteraciones del subproblema de inversión en la fase 3. Se inicia con *k* = 0.

1) Determinación de una buena solución inicial: *j* = 0

   Si *k* = 0 definir un J_t inicial constituido por las variables que pueden asumir un valor diferente de cero.

   Asumir la mejor incumbente de las soluciones anteriores como solución inicial y J_t es constituido por las variables diferentes de cero de esa incumbente. Si no existe incumbente factible definir un J_t inicial constituido por las variables que asumieron un valor diferente de cero en la solución óptima del problema anterior y ordenarlas de manera decreciente en relación a sus valores numéricos en la iteración anterior.

2) Prueba de Factibilidad:

   Si la solución parcial definida por J_t es factible entonces J_t es sondado; actualizar y almacenar la incumbente e ir al paso 10. En caso contrario ir al paso 3.

3) Prueba 1 de Balas:

Obtener el conjunto N_t^1. Si $N_t^1 = \emptyset$ entonces J_t es sondado e ir al paso 10. En caso contrario ir al paso 4.

4) Prueba 2 de Balas:

Obtener el conjunto N_t^2. Si $N_t^2 = \emptyset$ entonces J_t es sondado e ir al paso 10. En caso contrario ir al paso 5.

5) Prueba 3 de Balas:

Obtener el conjunto C_t. Si $C_t = \emptyset$ entonces J_t es sondado e ir al paso 10. En caso contrario ir al paso 6.

6) Prueba 3' de Glover-Zionts:

En las condiciones sugeridas por Glover-Zionts determinar si J_t puede ser sondado. Si J_t es sondado ir al paso 10. En caso contrario ir al paso 7.

7) Prueba de Ampliación de Geoffrion:

Si es posible ampliar J_t de acuerdo con la prueba de ampliación de Geoffrion entonces ampliar J_t hacer t → t + 1 y regresar al paso 2. En caso contrario, ir al paso 8.

8) Prueba de Ampliación de Glover-Zionts:

Si es posible ampliar J_t de acuerdo con la prueba de ampliación de Glover-Zionts, entonces ampliar J_t, hacer t → t + 1 y regresar al paso 2. En caso contrario ir al paso 9.

9) Prueba 4 de Balas:

Seleccione una variable libre para adicionar a J_t de acuerdo con la prueba 4 de Balas, t → t + 1. Si $v_{j^*} = 0$ entonces el nuevo J_t es factible. Actualice y almacene la incumbente, J_t es sondado e ir al paso 10.

En caso contrario regresar al paso 3.

10) Movimiento *Backward*:

Verificar si la enumeración implícita ya fue agotada $J_t = \emptyset$. Si la enumeración fue completada, fin, haga *k → (k + 1)* almacenar la solución óptima y verificar que no existan *sub-tours* en la solución actual.

En caso contrario hacer un movimiento *backward* de acuerdo con la prueba de Tuan, hacer $t \rightarrow (t + 1)$ y regresar al paso 2.

3.8.6 Metodología de identificación de sub-tours

El método de enumeración implícita 0-1 es una metodología de solución exacta usada para resolver problemas en la línea de la programación lineal entera, en donde las variables implicadas asumen valores binarios. La base de la metodología se encuentra basada en conceptos del *Branch & Bound* sin el uso de cuadros simplex como método de expansión y análisis en su espacio de solución. La eliminación del uso de cuadros simplex, es posible gracias a operaciones de adición y comparación en cada uno de los nodos, lo cuales permiten determinan la dirección del crecimiento del árbol y los cortes que deben realizarse para disminuir el espacio de búsqueda del método, de modo que sea posible descartar sub-espacios sin ser evaluados por no ser promisorios por factibilidad u optimalidad.

Sin embargo, al enfocar al método a la solución del problema TSP, es necesario usar un modelo completo del problema por la existencia de un conjunto de restricciones de *sub-tours* que asegura que no sean creados grupos de clientes desconectados de otros. Este grupo de restricciones además de ser fundamentales representan una gran carga en la matriz de restricciones del método. Esto impulso la determinación de una subrutina encargada de identificar grupos de clientes que forman *clusters* aislados del conjunto de clientes que deberían hacer parte de una ruta del VRP.

Como resultado del análisis del comportamiento de la metodología de enumeración implícita 0-1 se obtuvo que el algoritmo aseguraba un óptimo global si se incluían todas las restricciones de *sub-tours.* En pequeñas y medianas instancias el costo computacional resultó ser muy alto, por lo que se modificó la idea y se resolvió un modelo relajado, sin restricciones de *sub-tours*, y se construyó una sub-rutina de identificación de *sub-tours* que permitiera reducir el número de restricciones de *sub-tours* para que fueran las estrictamente necesarias en el problema. Con esto se logra un buen equilibro entre tiempo de procesamiento y respuestas de buena calidad.

La metodología propuesta se compone de dos fases que trabajan en conjunto para determinar una cantidad aceptable de restricciones de *sub-tours* que permita dar una solución efectiva al problema esclavo de rutas del VRP.

3.8.6.1 Determinación de restricciones de *sub-tours* por parejas de clientes

Como etapa de búsqueda de clústeres candidatos a generación de *sub-tours* en el problema, se propuso una subrutina encargada de recorrer el conjunto de clientes que formaran parte de la ruta del VRP que se requiere resolver. A partir de allí se realiza una etapa de agrupación en donde se analizan parejas de clientes alejados del conjunto de otros clientes. Esto da como resultado todas las posibles parejas de clientes que presentan gran separación respecto a los demás clientes de la ruta. Para esto se usa la distancia media entre todos los clientes del problema. Siguiente a este paso se supone que la pareja de clientes con mayor distancia es candidata formar un *sub-tour* de dos clientes. A continuación se crea esta restricción y se adiciona al problema. La idea principal es que una de las restricciones de *sub-tour* que más aparece es la formada por grupos de dos clientes. Para esto se siguen los siguientes pasos:

1. Se determinan todos los grupos de parejas de clientes en el problema.

2. Se determina la distancia media de todos los clientes y con base en esta, se plantea la búsqueda de la pareja más alejada de los clientes y se pasa a 3. En caso de no existir alguna pareja alejada, se termina la subrutina.

3. Se agregan las restricciones de *sub-tours* de cada uno de los clientes de la pareja alejada.

En la figura 3.5 se parte de los posibles clúster que pueden formarse por clientes próximos. De aquí se parte del cálculo de una distancia media entre todas las distancias entre clientes, la cual sirve por medida para la búsqueda de la pareja de clientes más alejada del todos los clúster, en este caso Cl_1. Finalmente, después de determinar el clúster más alejado, son la agregadas al modelo todas las restricciones de *sub-tour* correspondientes a los clientes que forman parte del clúster.

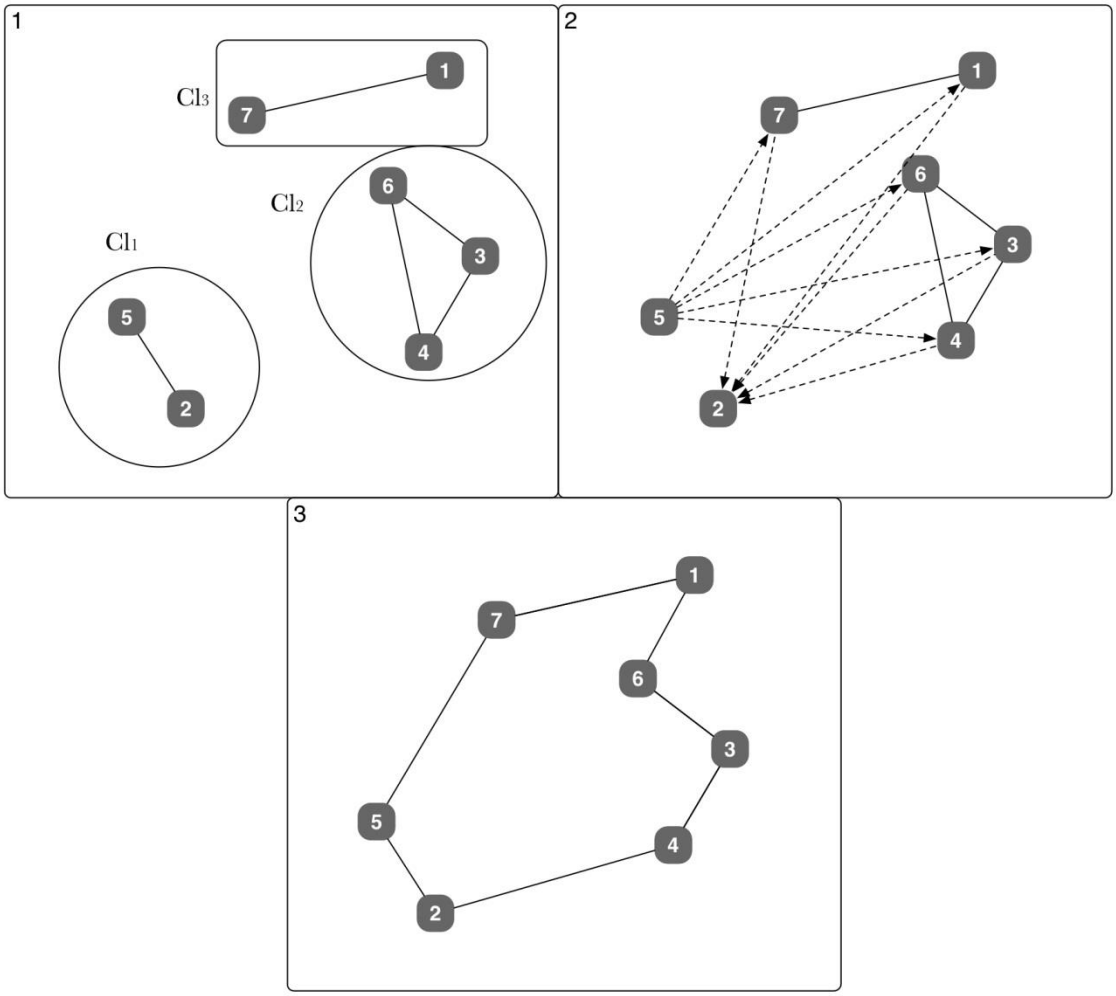


Figura 3.10 Determinación de restricciones de *sub-tour* por parejas de clientes

3.8.6.2 Determinación de restricciones de *sub-tours* por selección de clientes aleatorios.

La inexistencia de restricciones de *sub-tours* para los clientes en el modelo correspondiente a rutas individuales del VRP (que realmente corresponden a subproblemas TSP) normalmente produce convergencia a una respuesta con *sub-tours* de dos o tres clientes. Por tal motivo se creó un método que uniera parejas alejadas a grupos de clientes cercanos, lo cual permitía bajar el número total de restricciones de *sub-tours* llegando a respuestas con muy buenos resultados mejorando el uso de memoria y el tiempo de cómputo necesario para obtener las respuestas de cada TSP.

3.8.6.3 Conexión de clusters de clientes

Considerando que cada problema de la literatura especializada puede presentar variantes complejas, es posible encontrar casos en donde ninguna de las técnicas anteriores puede reparar por completo la aparición de *sub-tours* porque están formados por más de dos o tres clientes. En estos casos la metodología implementada usa una subrutina que conecta adecuadamente estos sub-tours a otros clientes próximos de la misma ruta. Esta técnica comienza su ejecución con la determinación de la conexión entre el *sub-tour* con el cliente más cercano que no pertenece al *sub-tour*. A continuación se determinan los clientes más atractivos del *sub-tour* para realizar la interconexión, lo cual implica la determinación de clientes extremos atractivos que presentan los mejores nodos a ser conectados con el otro grupo atractivo de clientes ya conectados al depósito o que también forman *sub-tours*. La conexión se realiza usando la filosofía del vecino más cercano adaptado a la ruta del VRP, dando como resultado la unión efectiva de los *sub-tours*.

En el ejemplo presentado en la figura 3.6 se observan tres clústeres de clientes diferentes de los cuales no se posee conexión alguna entre ellos. Para determinar estas conexiones en el paso 2 se buscan los bordes de cada uno de los clústeres, seguido en 3 de una etapa de búsqueda del clúster con el cliente más cercano al depósito, en este caso el cliente 4. Finalizando con el cálculo de la distancia entre los clientes pertenecientes a los bordes de cada clúster en el paso 4 y la unión de los más cortos que cumplieran con las restricciones de conexión planteadas para el TSP en la ecuación (2.9) En donde se asegura que cada cliente solo puede poseer una entrada y una salida.

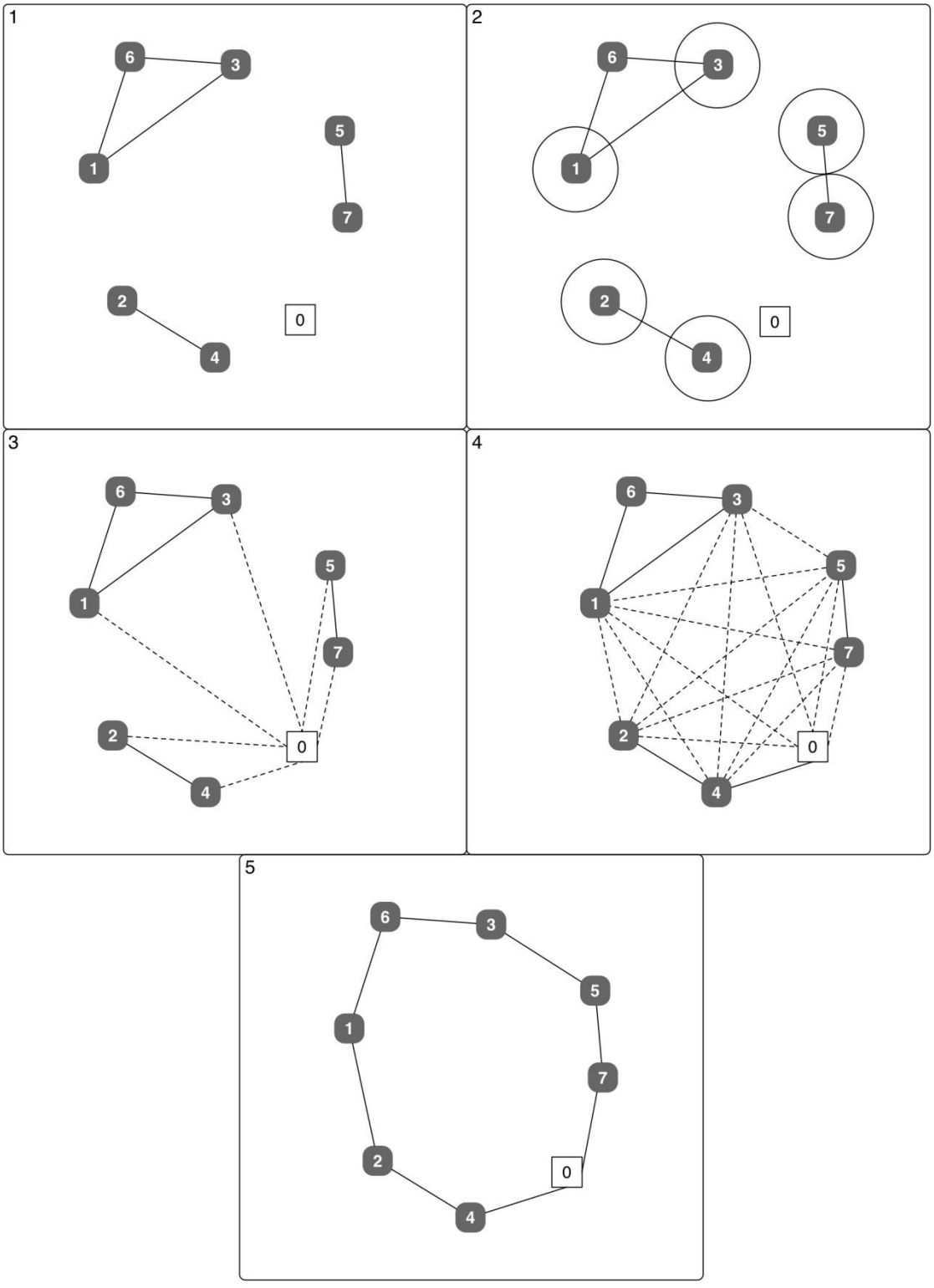


Figura 3.11 Unión de clústeres de clientes

3.9 Algoritmo general de la metodología propuesta

El algoritmo 1 presenta de manera general cada uno de los pasos de la metodología propuesta para solucionar el problema de ruteamiento de vehículos capacitado con flota homogénea, el cual se denomina Algoritmo genético de Chu-Beasley Modificado con Enumeración Implícita 1-0 (AGCBMEI).

$Datos$:
$Clientes \rightarrow n$
$Vehiculos \rightarrow k$
$Capacidad \rightarrow Q_k$
$Tamaño\ de\ la\ población \rightarrow N$
$Numero\ de\ indivduos\ en\ torneo \rightarrow S$
$Numero\ de\ mutaciones \rightarrow M$
$Rutas\ por\ secuencia \rightarrow R$
$Numero\ máximo\ de\ iteraciones \rightarrow MAX\_ITER$
$Población \rightarrow P$
$Valor\ fución\ objetivo \rightarrow FO$
$Solución\ incumbente \rightarrow Z_{min}$
$Mayor\ Función\ objetivo\ en\ la\ población \rightarrow Z_{max}$

$Resultados$:
$Solución\ problema\ CVRP\ con\ flota\ homogenea$

AGCBMEI:
$P \leftarrow \emptyset$
$Z_{max} \leftarrow \emptyset$
$Z_{min} \leftarrow \emptyset$
$countP \leftarrow 0$
$while$ ($countP\ != N$) **do**
    $P \leftarrow P \cup Secuencia\ (LKH), vecino\ más\ cercano, ahorro\ modificado;$
    $++countP$;
end
for ($i \leftarrow 1 \ldots N$) **do**
    $P \leftarrow P \cup División\ de\ cada\ secuencia\ mediante\ Q_k;$
end
$Z_{max} \leftarrow Determinar\ el\ peor\ individuo\ en\ P;$
$Z_{min} \leftarrow Determinar\ el\ mejor\ individuo\ en\ P;$
$while$ ($iter \leq MAX\_ITER$) **do**
    for ($Selec \leftarrow 1 \ldots S$) **do**
        $p_1, p_2 \leftarrow Selección\ por\ torneo\ en\ P;$
    $end\ for$
    $h \leftarrow Recombinación\ por\ rutas(p_1, p_2);$
    for ($mut \leftarrow 1 \ldots M$) **do**

$h \leftarrow Swap(rand_1, rand_2)$;
end for
for ($ruta \leftarrow 1 \ldots R(h)$) **do**
$h(ruta) \leftarrow$ *Ruta ordenada por* $Balas(ruta)$;
end for
if (*h cumple criterio de diversidad*) **then**
if($FO(h) < Z_{max}$) **then**
Ingresar h a P en cambio de Z_{max} ;
else
Rechazar h ;
end if
else
if ($FO(h) < Z_{min}$) **then**
Ingresar h a P en cambio de Z_{min} ;
else
Rechazar h ;
end if
end if
$Z_{max} \leftarrow$ *Actualizar el peor individuo en P*;
$Z_{min} \leftarrow$ *Actualizar el mejor individuo en P*;
$++iter$;
end while
return Z_{min} *en P*;

Algoritmo 1. Metodología AGCBMEI

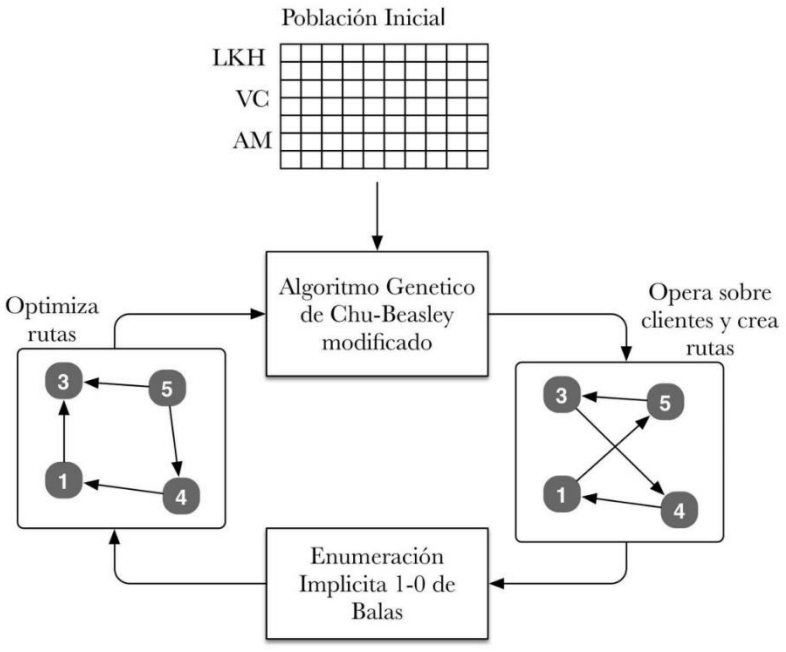


Figura 3.12. Representación abstracta de AGCBMEI

CAPÍTULO 4

PRUEBAS Y RESULTADOS

Como etapa final, se realizan pruebas computacionales para analizar el rendimiento, comportamiento y la contribución del algoritmo genético modificado con el método de numeración implícita 0-1 como etapa de mejoría, propuesto como técnica general de solución para el problema CVRP con flota homogénea. Se comparan los resultados en las instancias existentes en la literatura especializada a los cuales se les conoce la mejor respuesta hallada por otros autores usando diferentes algoritmos propuestos para el mismo problema.

La implementación de la metodología propuesta fue realizada en lenguaje C++ bajo el compilador G++ que hace parte de un conjunto de compiladores de licencia libre que es un derivado de la colección de compiladores GNU (GCC), apoyado en el editor de texto *SublimeText3*. El sistema operativo utilizado fue Ubuntu 14.04 con versión de kernel 3.14, en un computador con procesador Intel Core i5 a 3.2 GHz y memoria de 8 GB. A continuación se muestra una comparación entre las mejores soluciones obtenidas en la literatura especializada, para las instancias del CVRP, además de las diferentes implementaciones propuestas para el método de numeración implícita 0-1.

4.1 Resultados AGCBNI contra AGCB

En la tabla 4.1 se presenta una comparación entre el algoritmo genético de Chu-Beasley [4] modificado propuesto (AGCBM) y el mismo algoritmo genético propuesto con el método de enumeración implícita como etapa de mejoría. Ambos algoritmos fueron probados con las 27 instancias propuestas en [25] y se comparan con las mejores respuestas publicadas en la literatura especializada (*Best Known Solution* BKS) para medir su rendimiento y efectividad en la búsqueda de la respuesta óptima.

INSTANCIA	***n***	***r***	**BKS**	**AGCBMEI**	**GAP**	**AGCBM**	**GAP**
A-n32-k5	32	5	784	784	0	784	0
A-n33-k5	33	5	661	661	0	670	0,01362
A-n33-k6	33	6	742	742	0	742	0
A-n34-k5	34	5	778	778	0	805	0,0347
A-n36-k5	36	5	799	799	0	841	0,05257
A-n37-k5	37	5	669	669	0	710	0,06129
A-n37-k6	37	6	949	949	0	1121	0,18124
A-n38-k5	38	5	730	730	0	790	0,08219
A-n39-k5	39	5	822	822	0	900	0,09489
A-n39-k6	39	6	831	831	0	922	0,10951
A-n44-k6	44	6	937	970	0,035218783	1109	0,18356
A-n45-k6	45	6	944	964	0,021186441	997	0,05614
A-n45-k7	45	7	1146	1170	0,020942408	1270	0,1082
A-n46-k7	46	7	914	945	0,033916849	983	0,07549
A-n48-k7	48	7	1073	1125	0,048462255	1156	0,07735
A-n53-k7	53	7	1010	1094	0,083168317	1137	0,12574
A-n54-k7	54	7	1167	1227	0,051413882	1273	0,09083
A-n55-k9	55	9	1073	1155	0,076421249	1190	0,10904
A-n60-k9	60	9	1354	1420	0,048744461	1482	0,09453
A-n61-k9	61	9	1034	1165	0,126692456	1180	0,1412
A-n62-k8	62	8	1288	1422	0,104037267	1455	0,12966
A-n63-k9	63	9	1616	1750	0,082920792	1810	0,12005
A-n63-k10	63	10	1314	1504	0,144596651	1596	0,21461
A-n64-k9	64	9	1401	1576	0,124910778	1602	0,14347
A-n65-k9	65	9	1174	1375	0,17120954	1438	0,22487
A-n69-k9	69	9	1159	1296	0,118205349	1452	0,2528
A-n80-k10	80	10	1763	1978	0,12195122	2176	0,23426
				GAP_TOTAL:	0,052370322	**GAP_TOTAL:**	0,111549

Tabla 4.1. Tabla comparativa AGCBM y AGCBMEI

En las primeras tres columnas se muestra en orden, el nombre de la instancia utilizada seguido de la cantidad de clientes (*n*), la cantidad de rutas que componen la solución óptimo del problema (*r*), el BKS de cada problema seguido por los resultados obtenidos de las pruebas realizadas a los algoritmos AGCBM y AGCBMEI, acompañados del GAP entre la los BKS y la mejor solución alcanzada por cada metodología.

4.2 Resultados NIC contra NIRSD

Una de las motivaciones para continuar con la estrategia de reducción de restricciones de *sub-tours* para las rutas del problema VRP que se envían al subproblema de enumeración implícita, es la reducción del tiempo computacional para problemas de gran tamaño, lo cual es posible de observar en la tabla 4.2, que representa una comparación en tiempos de cómputo entre el método de enumeración implícita con el modelo completo que incluye las restricciones de *sub-tours* (NIC), y el mismo método de enumeración implícita con restricciones de *sub-tours* dinámicas (NIRSD), con *n* representando el número de clientes en cada problema que fue probado con ambos métodos.

PRUEBA	n	NIC	NIRSD	MEJORA
1	5	0,005 s	0,005 s	0,00%
2	6	0,089 s	0,086 s	-3,37%
3	7	1,006 s	0,942 s	-6,36%
4	8	2,038 s	1,938 s	-4,90%
5	9	26,689 s	12,541 s	-53,01%
6	10	58,907 s	35,714 s	-39,37%

Tabla 4.2 Tabla comparativo NIC y NIRSD

4.3 Conclusiones parciales sobre resultados

Al final del proyecto propuesto se encontraron una serie de resultados a través de las pruebas realizadas al método del algoritmo genético modificado apoyado en una técnica de enumeración implícita 0-1 como etapa de mejoría implementado para el problema de ruteo de vehículos capacitados con flota homogénea. De los resultados pueden obtenerse las siguientes observaciones:

- Los algoritmos genéticos son metodologias flexibles que pueden ser fácilmente adaptadas para resolver el problema VRP y sus variantes.

- Las características básicas de un algoritmo genético de Chu-Beasley [4] combinado con técnicas de recombinación por rutas y una etapa de mutación simple más un método exacto como etapa de mejoría, presenta un gran atractivo para la exploración del espacio de solución en el problema de ruteo de vehículos capacitados con flota homogénea (CVRP).

- En la etapa inicial, se presentó un procedimiento de búsqueda local por medio de heurísticas que permitieron obtener respuestas de muy buena calidad lo cual implicó una gran ventaja en la disminución de iteraciones necesarias para encontrar resultados de buena calidad para las instancias propuestas. La combinación de las tres estrategias heurísticas permite mantener una población inicial diversa evitando caer en soluciones sub-optimas de manera prematura.

- El cambio de la etapa de recombinación básica del método del algoritmo genético de Chu-Beasley por una recombinación por rutas, demostró ser una estrategia bastante eficiente que puede mantener características de las rutas obtenidas en la población inicial. La estrategia de intercambio de rutas, resulta atractivo al permitir ahorrar trabajo de reparación exhaustiva en el intercambio intra-rutas dado que muchas de estas rutas han sido resultado de un conjunto de heurísticas de buena calidad que disminuyen la necesidad de técnicas de mutación complejas, mejorando el rendimiento del algoritmo.

- El uso de una metodología exacta de enumeración implícita 0-1 permite explorar todo el vecindario de una ruta particular usando operaciones algebraicas simples. Esto puede constituirse en una alternativa interesante para problemas similares donde los problemas esclavos puedan ser definidos como problemas de programación binaria.

- La implementación del método de enumeración implícita 0-1 resulta interesante para determinar respuestas de buena calidad sin el uso de cuadros simplex, reduciendo la dependencia de librerías matemáticas especializadas o la creación de funciones específicas para el desarrollo de una rutina de solución exacta como pueden encontrarse actualmente en software profesional comercial como CPLEX, Gurobi, entre otros.

- Se puede observar que el método de enumeración implícita 1-0 de Balas permite mejorar la calidad de las soluciones respecto al algoritmo genético sin esta etapa de mejoría, sin embargo no alcanza la solución BKS reportada en sistemas de gran tamaño. Esto se atribuye al hecho de que el método exacto opera sobre rutas predefinidas y no tiene la posibilidad de eliminar clientes, adicionar clientes, crear nuevas rutas o eliminar rutas existentes. Este aspecto depende exclusivamente del algoritmo genético.

- Adicionar todas las restricciones de *sub-tours* resulta prohibitivo para el problema esclavo como se muestra en la tabla 4.2. Es más conveniente adicionar exclusivamente las restricciones de sub-tours necesarias.

- El algoritmo genético usado muestra gran potencial usado sin la etapa de mejoría. Los resultados se aproximan mucho a los que se obtienen con la etapa de mejoría.

- La técnica exacta utilizada es eficiente desde el punto de vista computacional y puede ser potenciada en trabajos posteriores para que pueda incluir clientes de las rutas vecinas.

- Se puede observar que en 10 de 27 instancias se obtiene la mejor solución conocida cuando utiliza el algoritmo genético de Chu-Beasley modificado en conjunto con la técnica de enumeración implícita 0-1 de Balas en la etapa de mejoría local.

CAPITULO 5

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

5.1 Conclusiones

- El algoritmo genético de Chu-Beasley es una metodología que entrega buenos resultados en el caso del problema CVRP. Una de sus principales desventajas es que requiere de una adecuada calibración de parámetros.

- Existe diferencia entre la calidad de las soluciones encontradas usando únicamente el algoritmo genético de Chu-Beasley y el algoritmo de Chu-Beasley con una etapa de mejoría exacta. Los resultados obtenidos son comparables con los mejores resultados publicados en la literatura especializada.

- Una de las razones por las que no se alcanza la mejor solución conocida en muchos casos es que deben plantearse mecanismos eficientes que permitan crear y destruir rutas en las soluciones parciales al igual que adicionar o retirar clientes de otra ruta para pasárselos a otra ruta.

- El uso de lenguajes de programación como C++ permite desarrollar algoritmos eficientes computacionalmente, es decir, con bajo consumo de tiempo de cómputo y de memoria.

- Al desarrollar una implementación para el algoritmo de enumeración implícita 1-0 de Balas se evita usar *solvers* comerciales para resolver los subproblemas de programación lineal entera. Al mismo tiempo se usa una metodología que consume poca memoria y que requiere bajos esfuerzos computacionales.

- El uso de heurísticas en el proceso de construcción de la población inicial del algoritmo genético permite reducir el tiempo de cómputo y mejorar la calidad de las soluciones encontradas.

5.2 Recomendaciones y trabajos futuros

- Al ser un algoritmo enfocado a la búsqueda de soluciones del CVRP con flota homogénea, es posible expandir la metodología como una subrutina de solución para problemas con múltiples depósitos de modo que esta pueda ser una etapa de determinación de rutas dependiendo de los clientes asignados en cada depósito, lo cual permitiría dar continuidad al algoritmo en problemas como el MDVRP y demás variantes del mismo.

- El uso del método de numeración implícita 0-1 adecua la búsqueda de soluciones por medio de operaciones matemáticas simples, lo cual es una gran ventaja al poder abrir las puertas de campo de la HPC (*High performance Computing*) por GPGPU (*General-Purpose Computing on Graphics Processing Units*) dado que los procesadores implicados en este tipo de hardware trabaja muy bien en la resolución de operaciones matemáticas básicas sin la necesidad del uso de librerías adicionales como CULA y CUBLAS que pueden hacer lenta la búsqueda en el espacio de solución.

- La falta de almacenamiento y el característico uso de una matriz A estática, permite que el paralelismo implícito del algoritmo pueda ser realizado para la eliminación de cuellos de botella presentes en la ejecución de problemas con grandes números de clientes y alta complejidad.

- El uso de técnicas de granularidad pueden ser agregadas a la mutación y recombinación para mejorar la búsqueda exhaustiva de la metodología propuesta.

BIBLIOGRAFÍA

[1] Anad G. D., Ramser J., “The truck dispatching problem,” Management Science, vol. 6, pp. 80–91, 1959.

[2] Balas E. "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", *Operations Research*, vol. 13, no. 4, pp.517 -546 1965

[3] Bodin L. .“A taxonomic Structure for Vehicle Routing and Scheduling Problems,”Computers and Urban Society. Vol 1,1974, pp 11-29.

[4] Chu E C. , Beasley J. E., "A genetic algorithm for the generalised assignment problem", Computers Ops Res. Vol. 24, No. 1, pp. 17-23, 1997

[5] Dantzing G., Ramser J. .“The Truck Dispatching problem”Managment Sciencie, vol6, No1.Octubre. 1959. pp.80-91.

[6] Escobar J. W. , Linfati R. , Toth P., “A two-phase hybrid heuristic algorithm for the capacitated location-routing problem,” Computers &OperationsResearch, vol. 40, pp. 70–79, 2013.

[7] Fusaka R., Longo H., Lysgaard v, De Aragão M. , Uchoa E., Reis M. , Werneck R. . “Robust BCP for The Capacitated Vehicle Routing Problem”. Math Program, Ser A. 2006. pp. 491-511.

[8] Gallego R.A., Escobar A.H., Toro E.M.: “Técnicas Metaheurísticas de Optimización”. Taller de publicaciones 2ª ed. Pereira (Colombia); Universidad Tecnológica de Pereira. 2008.

[9] Geoffrion A. M. "Integer Programming by Implicit Enumeration and Balas\' Method", *SIAM Review*, vol. 9, no. 2, 1967

[10] Glover F. , Zionts S. "A Note on the Additive Algorithm of Balas", *Operations Research*, vol. 13, no. 4, pp.546 -549 1965

[11] Glover F. "A Multiphase-Dual Algorithm for the Zero-One Integer Programming problem", *Operations Research*, vol. 13, pp.879 -919 1965

[12] Goldberg and D. , Lingle R. L. a., “The traveling salesman problem,” Proc. First Int. Conf. Genetic Algorithms and their Applications, pp. 154–159, 1985.

[13] Golden B. , Magnanti L., Nguyan H. . “Implementing Vehicle Routing Algorithms”. Networks,7(2),1972. pp. 113-148.

[14] HOLLAND, J. Adaptation in natural and artificial systems. Michigan: Ann Harbor, The University of Michigan Press, 1975.

[15] Laporte G. , Nobert Y. .“Exact alg.for the VRP". Discrete Mathematics. 31.1987. pp.147-184.

[16] Lin S. , Kernighan B. W., “An effective heuristic algorithm for the traveling-salesman problem,” Operations Research, vol. 21, pp. 498–516, 1973.

[17] Penna, P.H.V., Subramanian, A., Ochi, L.S. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem (2013) Journal of Heuristics, 19 (2), pp. 201-232.

[18] Pessoa A. , Uchoa E., De Aragao P.. The VRP: Latest advances and New Challenges". “Robust Branch-Cut-Price Algorithms for VRP”." Springer. 2008. pp. 297-325.

[19] Romero, R.; Monticelli, A., "A zero-one implicit enumeration method for optimizing investments in transmission expansion planning," *Power Systems, IEEE Transactions on* , vol.9, no.3, pp.1385,1391, Aug 1994

[20] Subramanian A., Penna P.H.V., Uchoa E., Ochi L.S.,"A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem",2012,"European Journal of Operational Research".

[21] Syswerda G., “Schedule optimization using genetic algorithms,” In A Handbook of Genetic Algorithms (Edited by L. Davis), pp. 332–349, 1991.

[22] Toth P. , Vigo D., The Vehicle Routing Problem, M. on Discrete Mathematics and Applications, Eds. SIAM, 2002.

[23] Tuan N. "A Flexible Tree-Search Method for Integer Programming Problems", *Operations Research*, vol. 19, no. 1, pp.115 -119 1971

[24] Tucker A. W. , Miller C. E., Zemlin R. A., "Integer Programming Formulation of Traveling Salesman Problems", Journal of the ACM (JACM), v.7 n.4, p.326-329, Oct. 1960

[25] http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/