UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

DESARROLLO DE UN ESQUEMA DE *BRANCH & PRICE* EN LA RESOLUCIÓN DE UN PROBLEMA DE RUTEO E INVENTARIO PARA UNA RED DE CAJEROS AUTOMÁTICOS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN GESTIÓN DE OPERACIONES

DANIEL GERARDO HERL CARRERO

PROFESORES GUÍA:
CRISTIÁN CORTÉS CARRILLO
PABLO A. REY

MIEMBROS DE LA COMISIÓN:
ALEJANDRO CATALDO CORNEJO
GUILLERMO DURÁN

SANTIAGO DE CHILE
2021

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN GESTIÓN DE OPERACIONES
POR: DANIEL GERARDO HERL CARRERO
FECHA: 2021
PROFESORES GUÍA: CRISTIÁN CORTÉS CARRILLO
PABLO A. REY

DESARROLLO DE UN ESQUEMA DE *BRANCH & PRICE* EN LA RESOLUCIÓN DE UN PROBLEMA DE RUTEO E INVENTARIO PARA UNA RED DE CAJEROS AUTOMÁTICOS

En esta tesis se propone el desarrollo de un esquema de *Branch & Price* en la resolución de un problema de ruteo e inventario multiperiodo para una red de cajeros automáticos. Las particularidades de este problema que lo hacen interesante son diversas y poco exploradas en la literatura especializada. Por una parte, en este problema se permite que el inventario de cajeros automáticos muestre quiebres de stock, generando dinámicas complejas de inventario. Adicionalmente, se permite la salida de vehículos desde el centro de distribución, para realizar sus recorridos, en un instante distinto al inicio de cada periodo, lo cual se traduce en un problema adicional de modelamiento complejo, ya que se deben generar filas y columnas al modelo de forma simultánea, lo cual es una particularidad que requiere de un tratamiento analítico especial en el esquema de generación de columnas propuesto como base del modelo *Branch & Price*. Adicionalmente, y como parte del esquema *Branch & Price*, se establece una estrategia de *Reliability Branching* así como una estrategia de selección de nodos ad-hoc al problema resultante. También se usa el concepto de *Farkas Pricining* para o bien hacer factible el problema o probar infactibilidad de los nodos y podarlos. Finalmente, se establece un esquema para seleccionar rutas iniciales que permitan comenzar el algoritmo a partir de un problema factible. A lo largo de la tesis se cubre en detalle la literatura necesaria para el modelamiento adecuado del problema y el desarrollo correcto e implementación del esquema de *Branch & Price*. Con esto se procede a resolver instancias del problema, en donde se identifican e implementan mejoras al modelo inicial y se identifican líneas de investigación futuras.

A mis 4 incondicionales,
María Betthy, Pedro, Paola y Natalie.

Agradecimientos

La realización y cierre de esta tesis no vino sin sus dificultades, tomó un arduo trabajo y cobró alto precio en mi persona. Este no hubiese sido posible sin un número no menor de personas que estuvieron allí para apoyarme y darme ánimo a seguir con ella, y es por esto que les agradezco.

En primer lugar agradezco a mis padres por su enorme apoyo y paciencia en mi trayecto académico. A mi hermana por su interés y preocupación. A mi novia Natalie por su amor, paciencia y comprensión en mis tiempos difíciles.

En segundo lugar a mis profesores guía, Cristián y Pablo, cuyos consejos y apoyo fueron instrumentales para finalizar este trabajo. A Alejandro (a.k.a sensei), a quién conozco desde pregrado en FEN, es imposible poder agradecer las oportunidades y la amistad que me ha extendido. Dentro del departamento de de transportes a Raúl por su invaluable ayuda al internarme en C++, su paciencia y disposición fueron de inmensa ayuda.

A mis amigos Hernán, Joaquín, Camilo y Daniel, quienes me han acompañado y dado aliento durante tantos años, en especial Daniel con quién he enfrentado ya 3 programas académicos, su amistad y apoyo ha sido invaluable. Finalmente me gustaría agradecer a todos quienes tuve la suerte de conocer durante la realización de mi tesis en el departamento de transporte, llenaron de alegría y amistad los momentos que pase allí.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

Dentro del mundo de los problemas de ruteo de vehículos, *vehicle routing problems* (VRP), podemos hallar los problemas de ruteo e inventario, *inventory and routing problem* (IRP), que tal como su nombre lo indica solucionan en conjunto el problema de administrar el inventario de los clientes que visita, los momentos en que deben ser visitados y la forma más eficiente posible de hacerlo, de modo de satisfacer las necesidades de sus clientes. Dentro de dicha área, las soluciones exactas no son muy comunes (Coelho et al., 2014) y entre ellas, el esquema de *Branch & Cut* es el más utilizado, siendo esquemas como *Branch & Bound*, *Branch & Price*, aún menos utilizados.

El esquema de *Branch & Price*, tal como se menciona anteriormente, es una generalización del esquema de *Branch & Bound*, que a diferencia de *Branch & Cut*, se enfoca en generación de columnas en vez de generación de filas (Barnhart et al., 1998). En la resolución del esquema de *Branch & Price* se permite la generación de columnas a lo largo del árbol del ramificación, ramificando cada nodo una vez que el algoritmo de generación de columnas no ha encontrado nuevas columnas para ser adicionadas al problema maestro del esquema.

El propósito de esta tesis es el desarrollar un esquema de *Branch & Price* para la resolución de un problema multiperiodo de ruteo e inventario. Este tiene como fin la correcta administración de los inventarios de una red de cajeros automáticos y los camiones que los abastecen, tomando en cuenta las condiciones particulares del problema que se desea resolver, descritas en el capítulo 2.

En el capítulo 1 se hace una revisión de la literatura relevante al área del problema que se desea resolver y al método de resolución a emplear. El capítulo se divide en tres partes, la primera realiza una revisión de los tipos de IRP, sus clasificaciones y los métodos que se han empleados para resolverlo, y finaliza con un resumen de la literatura asociada a problemas de ruteo e inventario de cajeros automáticos. La segunda parte cubre la teoría necesaria para desarrollar un esquema de *Branch & Price*, pasando por generación de columnas y *Branch & Bound*. La tercera parte realiza una descripción de SCIP (*Solving Constraint Integer Programs*), el software utilizado para programar (en C++) y resolver las instancias del problema.

El capítulo 2 describe las características particulares del problema que se desea resolver, como la dinámica de inventario particular en los cajeros automáticos y la facilidad dada a los camiones de iniciar las rutas con atraso, es decir, en un instante de tiempo distinto del inicio del periodo. Esta última particularidad fuerza la creación simultánea de filas y columnas en el esquema *Branch & Price*.

El capítulo 3 muestra y detalla en enfoque de *B&P* desarrollado para enfrentar este *IRP*. Específicamente, se detalla el modelo maestro desarrollado, el subproblema de generación de columnas junto con el esquema de *column-dependent-row* construido y las estrategias de ramificación y de selección de nodos utilizadas. El capítulo 4 describe las instancias solucionadas y los resultados de estas, se discuten e implementan modificaciones al esquema y las pruebas realizadas, sin embargo, se observan ciertas debilidades en la relajación lineal del problema maestro y un exceso en la generación de columnas.

Finalmente, las conclusiones cierran la tesis realizando un resumen de lo encontrado en los resultados, lo aprendido en el proceso y las posibles líneas de trabajo futuro.

Capítulo 1

Revisión Bibliográfica

El problema abordado puede fragmentarse en tres principales características. La primera corresponde a la clasificación del problema que enfrenta la tesis, un *Inventory and Routing Problem*. La segunda corresponde al método de resolución empleado, *Branch and Price*, derivado de la literatura de generación de columnas. Finalmente están las particularidades del problema, derivadas de los aspectos de este que lo separan de la literatura previa, como son los quiebres de stock (y los costos asociados a ellos) y la facilidad que se les da a los camiones para iniciar sus rutas en cualquier instante factible dentro de cada período y no únicamente al inicio de este. Dicha facilidad nos lleva a explorar la literatura de *column dependent rows* en donde la incorporación de nuevas columnas en el maestro provoca la inmediata creación de una o más restricciones pertinentes a la nueva variable.

Como aspecto relevante, y adicional, puede agregarse la herramienta utilizada para la programación y resolución del problema, esta es la librería SCIP (*Solving Constraint Integer Programs*) para **C++**, herramienta específica para programar y buscar soluciones exactas.

1.1. Inventory and Routing Problems

Este tipo de problemas es una extensión de la literatura sobre *Vehicle Routing Problems* (VRP), la cuál se puede seguir hasta el paper seminal de Dantzig and Ramser (1959) que rutea un set de camiones de carga de combustible que deben visitar un conjunto de estaciones de servicio. Los autores indican que este problema es una generalización del conocido problema del vendedor viajero (TSP, por sus siglas en inglés).

Tal como su nombre lo indica, los problemas de *Inventory and Routing* (IRP), tratan de resolver simultáneamente problemas de ruteo y de manejo de inventario en donde el administrador tiene que tomar al menos tres decisiones al mismo tiempo: (*i*) Cuándo visitar a cada cliente, (*ii*) cuánto entregar a cada cliente en cada visita realizada y (*iii*) cómo combinar estas decisiones de inventario con el ruteo de vehículos. Al incorporar estas decisiones en el proveedor el cliente deja de hacer pedidos, asumiendo el proveedor la responsabilidad de evitar quiebres de stock (caso en que no hay productos en inventario). Los beneficios para el

cliente están dados en el ahorro de costos de manejo de inventario y la confección rutinaria de pedidos, mientras que el proveedor puede incrementar el nivel de servicio en cada cliente y reducir sus costos de transporte al rutear eficientemente sus vehículos.

Tanto en Andersson et al. (2010) como en Coelho et al. (2014) se hace una exhaustiva revisión de la literatura referente a este tipo de problemas, el primero desde una perspectiva más industrial y de aplicaciones, mientras que el segundo tiene como fin de proponer una nueva clasificación enfocada en las variaciones estructurales posibles para el problema y la disponibilidad de información respecto a la demanda de los clientes que permita una mejor diferenciación entre modelos y algoritmos.

En Coelho et al. (2014) se identifican 7 variantes estructurales principales, cada una con dos o tres opciones posibles que los IRP pueden tomar. La Tabla 1.1 resume las diferentes opciones según variante:

Tabla 1.1: **Variantes Estructurales de IRPs**

Criterio	Opciones Posibles		
Horizonte de Tiempo	Finito	Infinito	
Estructura	One-to-One	Many-to-One	Many-to-Many
Ruteo	Directo	Multiple	Continuo
Política de Inventario	Maximum-level	Order-up-to level	
Decisiones de Inventario	Ventas Perdidas	Back Order	No negativas
Composición de Flota	Homogenea	Heterogenea	
Tamaño Flota	Único Vehículo	Múltiples Vehículos	Sin Cota

Fuente: Coelho et al. (2014)

Adicionalmente, los autores describen que la disponibilidad de información respecto a la demanda de los clientes y el cuando esta se hace disponible, es otra fuente relevante de diferenciación de los IRP revisados en su trabajo. Si la información está totalmente disponible de antemano el problema es determinístico, si se conoce su distribución el problema es estocástico y se tiene un *Stochastic Inventory and Routing Problem* (SIRP). Cuando no se tiene un panorama de la demanda, pero esta se va revelando en el tiempo, se posee un IRP dinámico. En este caso se puede ir explotando su distribución estadística a medida que la información se va haciendo disponible, lo cual nos entrega un IRP dinámico y estocástico (DSIRP por sus siglas en inglés).

Los primeros trabajos relacionados con IRP fueron mayoritariamente variaciones del problema de ruteo de vehiculos (VRP) que incorporaban nociones (o restricciones) de manejo de inventarios y heurísticas desarrolladas para tomar en cuenta los costos de estos. En Coelho et al. (2014) citan el trabajo de Bell et al. (1983) e indican que es un de los primeros trabajos en considerar en el problema de ruteo temas de inventario, en su resolución considera que debe cumplir con ciertos niveles mínimos y máximos de entrega para cada cliente y asigna un valor a cada unidad de producto entregada a los clientes.

Usualmente, los IRPs básicos se definen sobre un grafo $\mathbb{G} = \{V, A\}$, donde $V = \{0, 1, \ldots, n\}$ es el conjunto de nodos y $A = \{(i, j) : i, j \in V, i \neq j\}$ el conjunto de arcos. El nodo 0 se define como el depot o proveedor y el subconjunto $V' = V \setminus \{0\}$ representa el conjunto de

clientes o consumidores. Usualmente hay un costo de oportunidad o costo de inventario (*holding cost*) h_i por periodo ($i \in V$) y una capacidad máxima de almacenamiento de C_i. Existe un horizonte de planificación p y en cada periodo $t \in T = \{1, \ldots, p\}$ hay disponible un total de Q_t productos para repartir al set de clientes, que poseen una demanda d_i^t en cada periodo. Para el depot y cada cliente se define una variable I_i^t que indica el inventario en el nodo i al final del periodo t. Lo inventarios iniciales se definen como I_i^0 tanto para los clientes como para el depot. El set $K = \{1, \ldots, \mathbb{K}\}$ define el set de camiones a rutear y cuya capacidad esta dada por U_k. Finalmente se establece un costo c_{ij} de transporte asociado a utilizar el arco $(i, j) \in A$. Con esto la formulación del modelo básico queda:

$$\text{mín} \quad \sum_{(i,j)\in A} \sum_{t\in T} \sum_{k\in K} c_{ij} x_{ij}^{kt} \quad + \quad \sum_{i\in V} \sum_{t\in T} h_i I_i^t \tag{1.1}$$

Sujeto a

$$I_0^t = I_0^{t-1} + Q_t - \sum_{k\in K} \sum_{i\in V'} q_i^{kt} \qquad \forall t \in T \tag{1.2}$$

$$I_0^t \geq 0 \qquad \forall t \in T \tag{1.3}$$

$$I_i^t = I_i^{t-1} + \sum_{k\in K} q_i^{kt} - d_i^t \qquad \forall t \in T, i \in V' \tag{1.4}$$

$$I_i^t \geq 0 \qquad \forall i \in V', t \in T \tag{1.5}$$

$$I_i^t \leq C_i \qquad \forall i \in V, t \in T \tag{1.6}$$

$$\sum_{k\in K} q_i^{kt} \leq C_i - I_i^{t-1} \qquad \forall i \in V', t \in T \tag{1.7}$$

$$q_i^{kt} \geq C_i y_i^{kt} - I_i^{t-1} \qquad \forall i \in V', k \in K, t \in T \tag{1.8}$$

$$q_i^{kt} \leq C_i y_i^{kt} \qquad \forall i \in V', k \in K, t \in T \tag{1.9}$$

$$\sum_{i\in V'} q_i^{kt} \leq U_k y_0^{kt} \qquad \forall k \in K, t \in T \tag{1.10}$$

$$2y_i^{kt} = \sum_{j\in V} x_{ij}^{kt} + \sum_{j\in V} x_{ji}^{kt} \qquad \forall k \in K, t \in T, i \in V' \tag{1.11}$$

$$\sum_{\{(i,j)|i,j\in L\}} x_{ij}^{kt} \leq \sum_{i\in L} y_i^{kt} - y_l^{kt} \qquad \forall L \subseteq V', k \in K, l \in L \tag{1.12}$$

$$q_i^{rt} \geq 0 \qquad \forall i \in V', k \in K, t \in T \tag{1.13}$$

$$x_{ij}^{kt} \in \{0, 1\} \qquad \forall (i, j) \in A, k \in K, t \in T \tag{1.14}$$

$$y_i^{kt} \in \{0, 1\} \qquad \forall i \in V, k \in K, t \in T \tag{1.15}$$

En este modelo la variable x_{ij}^{kt} indica si el arco (i, j) fue utilizado por el camión k en el periodo t. Similarmente, la binaria y_i^{kt} indica si el cliente i es visitado por el camión k en el periodo t y, finalmente, la variable q_i^{kt} es la cantidad de producto entregado al cliente i en el periodo t por el camión k. En este problema se asumen una política de inventario *Up-to-Level* (OU).

Las restricciones (1.2) y (1.3) definen el funcionamiento del inventario en el depot, la primera determina cómo este va cambiando y la segunda previene quiebres de inventario. De similar forma funcionan las restricciones (1.4) y (1.5), que hablan de los inventarios en los clientes del problema. La restricciones (1.6) pone un máximo al inventario que un cliente o el depot pueden tener, mientras que (1.7) pone una cota superior a la cantidad que puede ser entregada a un cliente en cada periodo. Adicionalmente las restricciones (1.8) y (1.9), ponen como mínimo a entregar a un cliente la diferencia entre su capacidad máxima y el inventario del periodo anterior, si es que el cliente es visitado, alternativamente la segunda fija en cero la cantidad entregada a un cliente si es que no es visitado en tal periodo. La restricción (1.10) dice que ningún camión, si es usado en el periodo en cuestión, puede llevar más producto de lo que su capacidad indica.

Las restricciones (1.11) y (1.12) son restricciones del grado de cada nodo y de eliminación de sub-tours. Finalmente las restricciones (1.13) a (1.15) establecen la naturaleza de cada variable del problema básico. Este modelo básico de IRP es NP-Hard puesto que es una extensión del modelo clásico de VRP. Por esta razón muchas de las soluciones en la literatura son heurísticas, sin embargo, también hay un buen número de algoritmos de solución exacta.

Dada la complejidad del problema las primeras soluciones propuestas son heurísticas aplicadas en simplificaciones del modelo básico, por ejemplo Bell et al. (1983) sólo toma en cuenta los costos de transporte, mientras que Golden et al. (1984) y Dror and Ball (1987) optimizan sobre un horizonte de tiempo de un solo periodo, este último utiliza un algoritmo de asignación donde se asignan los clientes a fracciones del periodo para luego rutear tales asignaciones utilizando un algoritmo modificado de *Clarke & Wright*.

Otro enfoque de solución es la descomposición del problema en sub-problemas jerárquicos en donde la solución de uno de estos es input del siguiente sub-problema. En Campbell and Savelsbergh (2004) los autores descomponen el problema en dos fases, la primera asigna los clientes a días y a camiones, y se decide la cantidad a entregar tomando un horizonte de planificación de largo plazo. En la segunda fase se escoge el tiempo de salida de cada camión y el orden que deben seguir, sin embargo las cantidades estipuladas en la fase anterior pasan a ser sugerencias y pueden variar a forma de encajar en el plano de corto plazo, para el ruteo se utiliza el procedimiento GRASP (*Greedy randomized adaptative search procedure*). Otro trabajo que utiliza un método de descomposición es Cordeau et al. (2015) para un IRP multi-producto, en donde el problema es descompuesto en tres etapas. En la primera etapa se planifica los clientes que deben ser visitados en cada uno de los días del horizonte de planificación y las cantidades a repartir. En la segunda etapa se procede a rutear los camiones mediante una heurística de inserción, además se permiten *split-deliveries*. En la tercera, y última etapa, se procede a re-optimizar el problema utilizando un enfoque de rutas simples que incluyen el set de rutas obtenidas en la segunda etapa, rutas directas a cada cliente, un set de rutas simples obtenidas mediante un procedimiento de encadenamiento y una ruta que integra todos los clientes del problema.

En Bertazzi and Speranza (2012) se encuentra una descripción de las heurísticas para el tipo de problemas estudiados, llegando al más reciente tipo de estas, las matheurísticas, también llamadas heurísticas híbridas o heurísticas basadas en optimización, que incluyen modelos de programación matemática (usualmente problemas lineales de programación entera

mixta, MILP) dentro de esquemas de resolución heurísticos o metaheurísticos con el fin de explotar el poder de los actuales software de optimización (comerciales o no). Algunas de ellas alcanzan excelentes resultados, inclusive encontrando varios óptimos como es el caso de Archetti et al. (2012) donde proponen una nueva heurística mixta de Tabú Search en conjunto de dos problemas enteros utilizados para la intensificación de la búsqueda local, una vez que se ha encontrado una nueva mejor solución.

Usualmente, los MILP son utilizados para tareas específicas dentro de estos esquemas. Estas tareas pueden ser:

i. La resolución de sub-problemas.

ii. Resolución de sub-partes de una instancia.

iii. Restringir el espacio de búsqueda de soluciones.

iv. La exploración de vecindarios de soluciones.

Dentro del grupo (*i*) podemos encontrar las matheurísticas basadas en ruteo y aquellas que resuelven el problema de inventario y volumen de entrega primero y luego el de ruteo. En la primera usualmente se pasan por alto los costos de inventario puestos que estos no son directamente pagados, por tanto son fácilmente olvidados, y debido a la amplia cantidad de métodos de resolución del VRP. En la segunda matheurística se descompone el problema en inventario (definiendo cúando se debe visitar y cuánto entregar), que es resuelto con un MILP, y el posterior problema de ruteo, en el que suele aplicarse una heurística, que debe resolverse en cada período dentro del horizonte de planificación.

En el grupo (*ii*) podemos encontrar las matheurísticas en donde los clientes son clusterizados en una primera etapa para luego, en cada cluster, aplicar un problema de programación entera. Según Bertazzi and Speranza (2012) esta metodología es común entre quienes resuelven IRPs cotidianamente, pero se advierte que se deben evitar configuraciones de clusters que hagan el problema infactible tal como lo demuestran los autores en su trabajo. En Bertazzi et al. (2019) es propuesta una matheurística de tres etapas para resolver una extensión del modelo básico en donde se contemplan no uno sino varios depot. En la primera etapa del enfoque se construyen sets de clusters tomando en cuenta múltiples características de los clientes, en la segunda etapa se crean un set de rutas intra e inter-clusters que toman en cuenta tanto la capacidad de los vehículos como el número máximo de clientes que pueden ser visitados por un depot y, finalmente, en la tercera etapa se resuelven un MILP basado en rutas para generar soluciones factibles (resuelto vía *Branch & Cut*).

En (*iii*) podemos encontrar las matheurísticas que reducen el espacio de soluciones al imponer restricciones o políticas, una de las más empleadas es, por ejemplo, imponer la política *Order-up-to* bajo la cuál el monto entregado, de haber una visita, es igual a la diferencia entre la máxima capacidad de inventario en el cliente y su actual nivel de inventario. En consecuencia, las decisiones de cúando visitar definen automáticamente la cantidad a entregar. Por otra parte, dentro de grupo (*iv*) donde es posible ocupar MILP para intensificar la búsqueda de mejores soluciones dentro del vecindario de una solución previamente encontrada. Tal es el caso de Archetti et al. (2012) donde se define una nueva matheurística que combina Tabú

Search con dos sub-problemas MILP que intensifican búsquedas locales, la cuál llaman *Hybrid Approach to Inventory Routing* (HAIR). Los resultados del enfoque muestran que los gap al óptimo se encuentran sistemáticamente bajo el 1 %, cuando el óptimo puede ser encontrado, y también llega a un buen porcentaje de óptimos para varias instancias *benchmark*.

Por otra parte están las soluciones exactas, aquellas que apuntan a encontrar soluciones óptimas al problema planteado. En Archetti et al. (2007) se desarrolla el primer algoritmo de *Branch & Cut* (B&C). En este los autores plantean un modelo MILP en donde se considera una política *Up-to-Order* y solo un vehículo. Para fortalecer la formulación y mejorar la cota inferior a través del esquema de *Branch & Bound*, en cada nodo se agregan dinámicamente una serie de *valid inequalities* (restricciones válidas). Adicionalmente, restricciones de eliminación de sub-tours se agregan a lo largo del árbol a medida que alguna de las soluciones incumbentes las violan. Las pruebas realizadas con este esquema incluyen instancias generadas aleatoriamente con hasta 50 clientes y 6 períodos. Finalmente los autores comparan los resultados del problema resuelto cambiando la política de inventario a una de *Maximum Level* y a otra de valor libre, las comparaciones indican que relajar la política de UO permite obtener mejores resultados, en especial en instancias de mayor tamaño y horizonte de tiempo, sin embargo cuando la cantidad a entregar se deja libre los inventarios en cada cliente pueden aumentar significativamente hasta puntos inaceptables.

Leandro Coelho y Gilbert Laporte han experimentado a partir del trabajo de Archetti. En Coelho and Laporte (2013a), los autores expanden el trabajo de Archetti et al. (2007) desarrollando un *Branch & Cut* para un IRP multi-vehículo y multi-producto en el cual resuelven para múltiples instancias y en donde prueban políticas de regularidad, las cuales indican que debe haber un cierto espacio de tiempo entre las visitas a un mismo cliente y que la mayor parte de las visitas a cada cliente deben ser realizadas por un mismo vehículo (conductor). Por otra parte en Coelho and Laporte (2013b) resuelven distintos clases de IRPs, incorporando flota heterogénea, traspaso de inventario entre clientes (*transshipment*) y la opción resuelta en el trabajo anteriormente mencionado (con políticas de regularidad).

Posteriormente, en Coelho and Laporte (2014), los autores resuelven un IRP al cual agregan nuevas restricciones válidas relacionadas a la demanda de los clientes y la capacidad de los vehículos de la flota, adicionalmente experimentan con el orden en que se arman las rutas, con la idea de que al demandar que se visiten ciertos clientes primero exista menos flexibilidad con la que extender la búsqueda de soluciones y el algoritmo de *Branch & Cut* itere menos y llegue a un resultado en menor tiempo. Las restricciones válidas propuestas indican que debe haber una visita en un horizonte de tiempo $[t_1, t_2]$ si es que la demanda en dicho horizonte es mayor a la cantidad de inventario que se puede almacenar, también se agrega la restricción que indica que un cliente debe ser visitado si es que la demanda del horizonte es mayor al inventario al inicio de dicho periodo. Por otro lado el *input ordering* con el que experimentan los autores se utiliza explotando las restricciones de simetría, que plantean que el vehículo k no puede ser utilizado si es que el vehículo $k-1$ tampoco y que si el cliente i es asignado al vehículo k en el período t entonces el vehículo $k-1$ sólo puede atender clientes con un índice menor a i, entonces al estar ordenados los clientes estos se van visitando en dicho orden. Experimentan asignando menores índices a clientes d mayor demanda, de forma de que los vehículos se llenen más rápido y se obligue a utilizar un segundo vehículo (mejorando la cota inferior en el proceso). Además de este orden, se experimenta ordenando los

clientes por lejanía y por proximidad al depot, finalmente como comparación se evalúa un orden aleatorio de los clientes. Los resultados indican que ordenar por demanda y por lejanía al depot reportan mejores resultados en términos de tiempo de resolución y cotas promedios (aumento de inferior y reducción de la superior), estos resultados se comparan con los de Archetti et al. (2007) mostrando mejoras moderadas.

El algoritmo *Branch & Price* también ha sido utilizado para encontrar soluciones exactas a IRPs. Es el caso de Grønhaug et al. (2010), en el cual se resuelve un IRP para el transporte de Gas Natural Condensado (LNG, en inglés) entre las plantas de producción que condensan el gas para su transporte y los puertos de evaporación (gasificación) en donde se vuelve a transformar en gas para su almacenamiento y venta. Los autores modelan el problema tal que el maestro contiene las restricciones de inventario y capacidad de todos los puertos (tanto de los de condensación como los de gasificación) mientras que el subproblema de generación de columnas se encarga de generar columnas (rutas) para los buques. Por lo demás el subproblema es resuelto vía un algoritmo ad-hoc de programación dinámica que compara rutas parciales sólo en nodos de carga y descarga, es necesario hacer modificaciones puesto que en el transporte de LNG una fracción de la carga se gasifica y es utilizada como combustible para el buque. Otros enfoques utilizados para problemas de transporte marítimos con manejo de inventario han sido: generación de columnas (ver Christiansen (1999)) y la enumeración de caminos en conjunto de *Branch & Bound*. Por otro lado en Grønhaug and Christiansen (2009) encontramos el primer problema LNG-IRP resuelto vía enumeración de caminos. El trabajo de Grønhaug et al. (2010) logra mejores resultados que Grønhaug and Christiansen (2009) en menor tiempo, sin embargo todavía falta demostrar optimalidad para varias de las instancias probadas.

Uniendo los enfoques de generación de columnas con la planos cortantes tenemos el algoritmo de *Branch-Price-and-Cut*, acá podemos encontrar el trabajo de Desaulniers et al. (2016) que actualmente es la mejor solución exacta genérica. La formulación hace uso de patrones de rutas de entrega (RDP en inglés) desarrollados en Desaulniers (2010), y variables de decisión continuas que indican la proporción de una ruta operada bajo una RDP. En dicho trabajo se hace uso de un algoritmo de etiquetas para resolver un *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC) en el subproblema de generación de columnas, en generación de cortes se evaluan restricciones que apuntan al mínimo número de visitas por cliente, mínimo número de visitas por intervalo de tiempo y restricciones de capacidad adaptadas a un IRP del tipo usadas en Laporte et al. (1985). El algoritmo diseñado se utilizó sobre un set de 640 instancias de *benchmark*, sobre las cuales se demostró gran rendimiento, superando a los conocidos algoritmos de *B&C* (Archetti et al. (2007), por ejemplo) para instancias con cuatro y cinco vehículos. Inclusive, de las 238 instancias sin resolver en el set se pudo probar optimalidad para 54 de ellas.

1.1.1. Automated Teller Machines

Al revisar la literatura en IRP relacionados a cajeros automáticos encontramos los trabajos de Van Anholt et al. (2015), Larrain et al. (2017), Chotayakul et al. (2013) y dos más, Koc et al. (2018) y Batl and Gözüpek (2019). Ninguna de las cinco publicaciones trabaja con demanda estocástica, esta se asume conocida para todo el periodo de tiempo planificado. Por

lo demás, sólo una de ellas extiende la posibilidad de *stock-outs* en su formulación (Larrain et al., 2017), mientras que para el resto no hay opción de que los cajeros no cumplan con la demanda de cada uno de los períodos.

En Van Anholt et al. (2015) encontramos un IRP con *pick up-and-delivery* para una red de cajeros automáticos que no sólo hacen giros de efectivo sino también aceptan depósitos (RATM, por sus siglas en inglés). Si bien problema no permite *stock-outs* este sí permite que los camiones permanezcan un tiempo adicional en ruta más allá de la duración máxima que puede tener cada ruta, con un costo asociado por minuto de exceso.

Los autores desarrollan una formulación basada en arcos la cual resuelven vía *Branch & Cut*, las restricciones de eliminación de subtours se van generando dinámicamente hasta que se encuentra una solución factible o no se pueden agregar más restricciones, posteriormente empieza el *branching* en variables con decimales.

Para atacar instancias reales el trabajo ejecuta un algoritmo de clusterización previa en cada período que clasifica los cajeros en 5 grupos: Aquellos en los que hay que recoger dinero, aquellos en los que hay que entregar dinero, los que quizás hay que entregarles, los que en los que quizás hay que retirar efectivo y, finalmente, el grupo de los que no deben ser visitados. La idea de generar estos clusters en cada período dentro del horizonte de tiempo es reducir significativamente el tiempo de resolución de algoritmo de *B&C* puesto de que varias de las decisiones ya son realizadas por la clusterización.

Por otra parte en Larrain et al. (2017) encontramos un IRP para la administración de una red de cajeros automáticos en Santiago de Chile. En dicho trabajo, y a diferencia del anteriormente descrito, se permiten *stock-outs* en los cajeros pero se prohíbe que las rutas excedan el tiempo máximo estipulado. Los *stock-outs* son penalizados en proporción a la cantidad de demanda del período no satisfecha. Otra particularidad del trabajo es la forma en que los cajeros son rellenados puesto que sólo se puede entregar efectivo en *cassettes* (contenedores con un monto fijo de dinero). Al visitar un cajero se intercambian *cassettes* y el valor del inventario pasa a ser el valor del *cassettes* entregado, mientras que el vehículo se lleva consigo el *cassettes* antiguo, con cualquiera sea la cantidad de efectivo remanente en él.

El problema es modelado como un MIP cuya formulación se basa en arcos, esta es reforzada con desigualdades válidas como las encontradas en Archetti et al. (2007), Coelho and Laporte (2014) y Coelho and Laporte (2013a). La resolución de este problema se realiza utilizando un algoritmo de *B&C* en conjunto con un esquema, que los autores llaman, *Variable MIP Neighborhood Search* (VMNS). Este esquema realiza búsquedas locales en base a una solución factible encontrada por el algoritmo de *B&C*, específicamente divide la solución en vecindarios donde cada uno de ellos es una ruta de la solución factible, posteriormente para cada ruta (vecindario) soluciona un subproblema similar al problema inicial pero en donde todas las variables de rutas y cantidades están fijas (con valores iguales al de la solución inicial) excepto las de la ruta en cuestión. Este subproblema es nuevamente resuelto vía *B&C*, si se encuentra una mejor solución factible esta es entregada al algoritmo original de *B&C*, de otra forma se procede a hacer una búsqueda local en el siguiente vecindario. Este esquema de solución permite a los autores resolver instancias de hasta 60 cajeros, sin embargo las soluciones se deterioran con rapidez a medida de que se toman en cuenta más periodos ya que no es posible encontrar soluciones en el tiempo máximo estipulado (3 horas).

En el trabajo de Koc et al. (2018) se modela y resuelve el problema del manejo de una red de cajeros automáticos en Gaziantep, Turquía. En este los autores consideran una red multi-depot con *pick up-and-delivery* y una flota de vehículos homogénea, sin embargo los autores no consideran en su función objetivo el costo de inventario en los ATMs ni permiten *stock-outs* en ellos, tal como en los ejemplos anteriores la formulación es basada en arcos, pero el método empleado para la resolución del problema es un *Tabú Search* en donde la fase de mejora (*improvement phase*) de las soluciones encontradas incluye el cambio de orden de los cajeros en la ruta, el cambio de un cajero a otra ruta o el intercambio de cajeros entre rutas.

Alternativamente, en Chotayakul et al. (2013) determinan cuánto dinero entregar en sucursales bancarias y ATMs en cada periodo de tiempo. El problema es formulado como un MIP, sin embargo este es reformulado pensando en ser resuelto como un problema de rutas más cortas con el cual poder encontrar resultados cercanos al óptimo. En Batl and Gözüpek (2019) se abre la posibilidad de, en conjunto con tomar las decisiones de usuales de un IRP, decidir qué cajeros de la red reemplazar por RATM, la formulación empleada es resuelta por un algoritmo *ad-hoc* al problema diseñado por los autores en donde las decisiones se van tomando por separado. Tomando en cuenta un horizonte de tiempo de siete días se calculan los días en que cada cada cajero debe ser visitado y el monto a entregar minimizando el costo de vista y el costo alternativo del dinero. Posteriormente para cada camión se construyen rutas con potenciales cajeros a visitar y en la tercera etapa de la heurística finalmente se asignan, el método termina chequeando las restricciones de capacidad y fraccionando las cantidades a repartir entre las rutas en caso de que alguna la incumpla.

1.2. Generación de Columnas

Esta sección se extiende sobre el método de generación de columnas y las extensiones asociadas a este, que son relevantes para este trabajo. Este método nace principalmente con la idea de poder abordar problemas con un gran número de variables, como es el caso de los problemas de ruteo como se comenta en Feillet (2010). Lo que se hace es descomponer la formulación en un problema maestro y un subproblema de generación de columnas que confecciona columnas para el maestro, que beneficien la función objetivo del primer problema. Se entiende como columna el vector de parámetros asociados a una variable específica del problema maestro, es decir, el subproblema encuentra variables, y sus parámetros, dentro del conjunto de variables ausentes del problema maestro (Bertsimas and Tsitsiklis, 1998).

Para entender mejor, consideremos el siguiente problema lineal,

$$z_{MP} := \min \sum_{i \in \Omega} c_i x_i \tag{1.16}$$

$$\text{s.a}$$

$$\sum_{i \in \Omega} a_i x_i \geq b \tag{1.17}$$

$$x_i \geq 0 \quad \forall i \in \Omega \tag{1.18}$$

Donde $b \in \mathbb{R}^m$, $a_i, c_i, x_i \in \mathbb{R}$ para $i \in \Omega$. Nos referiremos a este problema como *problema maestro* ($MP(\Omega)$), con un valor óptimo z^*_{MP}. En este problema tendremos que el número de elementos del set Ω (que suponemos finito) es mucho mayor que el número de restricciones o filas, m, y para cada cada uno de sus elementos existe una variable definida en el $MP(\Omega)$. Asociados a cada variable x_i existen los parámetros a_i y c_i, en consecuencia los vectores $[c_i, a_i]^T \, \forall i \in \Omega$ son las columnas del problema maestro, donde cada columna esta asociada a una de sus variables. Luego, en cada iteración del algoritmo de simplex se busca una variable no básica que pueda entrar a la base, una que posea costos reducidos negativos, es decir la variable $i \in \Omega$ que minimice $\bar{c} := c_i - \lambda^T a_i$, donde λ es el vector de variables duales del problema maestro.

La idea detrás de generación de columnas es que el set de elementos en Ω es demasiado grande como para poder enumerar la totalidad de variables (y sus columnas asociadas), y en consecuencia el problema es demasiado grande como para ser resuelto con todas sus variables en él al mismo tiempo. Por lo demás en la solución óptima (y en la mayoría de las soluciones factible) del problema la mayor parte de las variables será no básica, y por tanto, estarán fijas en cero. En consecuencia, para resolver el problema sólo necesitamos un subset $\Omega' \subset \Omega$ de variables. Al sustituir el set Ω con el subset Ω' en el problema maestro obtenemos al denominado problema maestro restringido (RMP o $MP(\Omega')$):

$$z_{RMP} := \text{mín} \sum_{i \in \Omega'} c_i x_i \tag{1.19}$$

$$\text{s.a}$$

$$\sum_{i \in \Omega'} a_i x_i \geq b \tag{1.20}$$

$$x_i \geq 0 \quad \forall i \in \Omega' \tag{1.21}$$

Como se puede observar la única diferencia es el set de variables entregadas al MP. En el óptimo del $MP(\Omega')$ tendremos que su valor objetivo es z^*_{RMP} y su solución es el vector $[x^T_{RMP}, \left(0^{|\Omega \setminus \Omega'|}\right)^T]$, que por simpleza denotaremos como x^*_{RMP}. Llamaremos λ^*_{RMP} y λ^*_{MP} a las soluciones duales asociadas a los problemas duales de RMP y de MP, respectivamente, con valores objetivos $z^*_{D(\Omega')}$ y $z^*_{D(\Omega)}$. En generación de columnas el criterio de parada utilizado para asegurar que $z^*_{RMP} = z^*_{MP}$ es que la solución dual λ^*_{RMP} sea factible en $D(\Omega)$. Para ver lo anterior detallemos que la solución de $D(\Omega')$ también define una solución, no necesariamente factible, de $D(\Omega)$, dado que ambos problemas están compuestos por las mismas restricciones, ergo, $D(\Omega')$ es una relajación de $D(\Omega)$, y como ambos son problemas de maximización, tendremos que $z^*_{D(\Omega)} \leq z^*_{D(\Omega')}$. En consecuencia la condición de que λ^*_{RMP} sea factible para $D(\Omega)$ asegura que $z^*_{D(\Omega')} \leq z^*_{D(\Omega)}$, es decir, en el óptimo tendremos que $z^*_{D(\Omega)} = z^*_{D(\Omega')}$ y $z^*_{MP} = z^*_{RMP}$ (por dualidad fuerte, revisar Bertsimas and Tsitsiklis (1998)). Por lo demás, los valores z^*_{RMP} que se van encontrando en el camino a la convergencia al óptimo se utilizan como cotas superiores del valor óptimo del $MP(\Omega)$.

A partir de lo anterior, el método de generación de columnas va expandiendo y/o modificando el set Ω a medida que encuentra restricciones violadas en el problema dual (variables del MP con costo reducido negativo). Es decir, el método en cierto sentido reproduce la lógica

del algoritmo de simplex, sólo que gran parte de las variable no básicas no están en el $MP(\Omega')$ y el chequeo y generación de nuevas variable queda en manos de un subproblema. Notar que, puesto que el set Ω es finito, el método finalizará en un número finito de iteraciones. Como se dijo anteriormente es el subproblema el encargado de encontrar nuevas variables dentro del set $\Omega \setminus \Omega'$, para que la resolución, alternadamente, entre el $MP(\Omega')$ y el subproblema permita la resolución del problema maestro, lo anterior se describe en el algoritmo (1).

En el ejemplo expuesto anteriormente, el subproblema constaría de encontrar restricciones violadas en el problema dual del $MP(\Omega)$ utilizando la fórmula de costos reducidos:

$$\overline{c}^* := \min_{c_i}\{c_i - \lambda^T a_i\} \tag{1.22}$$

Donde λ^T es el vector de variables duales que representan las restricciones (1.20). Sin embargo, observando la expresión (1.22), esta no es diferente de atacar el problema maestro con todas sus variables. Esto puede corregirse al darnos cuenta de que si bien los elementos de Ω no se encuentran explícitamente enumerados, sino que se manifiestan implícitamente como puntos factibles $h \in H$ de un problema de optimización, tal que podemos reescribir la expresión (1.22) como:

$$\overline{c}^* := \min_{h \in H}\{c(h) - \lambda^T h\} \tag{1.23}$$

Donde el costo de la variable es función del punto h encontrado. Ahora, si el resultado de este subproblema de optimización cumple con la condición $\overline{c}^* < 0$ entonces la columna encontrada, y su variable asociada, ingresan al $MP(\Omega')$.

Algorithm 1: Algoritmo Genérico de Generación de Columnas

Input: Set inicial de variables en Ω'

1 Inicializar $MP(\Omega')$, con sus respectivos parámetros.
2 Resolver $MP(\Omega')$ y obtengo z^*_{RMP}
3 $\lambda_{MP} \longleftarrow$ Obtengo valores duales del $MP(\Omega')$
4 Resolver subproblema $\longrightarrow \overline{c}^*$
5 $\kappa \longleftarrow$ Obtengo columna(s) desde el subproblema
6 **if** $\overline{c}^* < 0$ **then**
7 | $\Omega' \longleftarrow \Omega' \cup \kappa$
8 | Volver a paso 1.
9 **else**
10 | **return** z^*_{RMP}
11 **end**

Finalmente, el valor objetivo del $MP(\Omega')$ en conjunto con $\overline{c}^*$ pueden ser utilizados para derivar una cota inferior para el problema maestro. En Desrosiers and Lübbecke (2005) muestran que dado un valor $\theta \geq \sum_{i \in \Omega} x_i$ en la solución actual del RMP, el valor z^*_{RMP} no

puede ser reducido más de θ veces el valor del menor costo reducido, $\overline{c}^*$. En consecuencia se puede estimar una cota inferior de la siguiente forma:

$$z^*_{RMP} + \theta\overline{c}^* \leq z^*_{MP} \tag{1.24}$$

En el caso en que el problema en mano sea un MIP o un ILP, el RMP se relaja y las variables enteras se tratan como continuas. En el momento en que el algoritmo de generación de columnas se detiene y ha encontrado el óptimo de la relajación lineal, el RMP relajado y el set de variables que dejó el subproblema son entregados al algoritmo de *Branch & Bound* (*B&B*), el cuál ejecuta un búsqueda dentro de las posibles soluciones enteras y devuelve aquella de mejor valor objetivo.

1.2.1. Branch & Bound

Branch & Bound es un esquema ampliamente utilizado para encontrar soluciones exactas para una gran variedad de tipos de problemas de optimización. La principal idea del esquema es dividir sucesivamente la instancia del problema en subproblemas más pequeños que son más fáciles de resolver. El mejor resultado entre todos los subproblemas es el óptimo global del problema. Implícitamente lo que hace es enumerar todas la posibles soluciones al problema sin tener que efectivamente hacerlo, esto gracias a las estrategias de *branching*, selección de subproblemas y poda, que van guiando la búsqueda y eliminando regiones de soluciones incapaces de entregar una mejor solución.

La acción de dividir el problema en dos o más subproblemas es llamado *branching* (ramificar). En la ejecución del esquema se genera un árbol de ramificación (*branching tree*) en donde cada uno de sus nodos representa un subproblema como el que se puede observar en la figura (1.1). Se le llama nodo raíz (*root node*) al primer nodo del problema inicial P, mientras que las hojas de este son subproblemas que o bien son más fáciles de resolver o son subproblemas en el set $\mathbb{S}$ que por el momento no han sido evaluados. El algoritmo (2) describe el esquema de *B&B* para el siguiente problema genérico de minimización:

$$\begin{aligned} \text{mín} \quad & c^T x & & (1.25)\\ & \text{s.a} & \\ Ax \quad & \leq \quad b & & (1.26)\\ x \quad & \in \quad \{0,1\}^n & & (1.27) \end{aligned}$$

Donde $c \in \mathbb{R}^n$ y $A \in \mathbb{R}^{m\times n}$. Para efectos prácticos se denota X_{MIP} como el set de soluciones enteras factibles al problema, y al iniciar el esquema se establece que $X_{MIP} = \emptyset$ y $c^* = \infty$. En el sexto paso del algoritmo (2), llamado *bounding* (acotamiento), permite evitar la completa enumeración de las soluciones en X_{MIP}, las cuales pueden ser exponencialmente crecientes con el tamaño del problema, cuando un subproblema puede ser ignorado debido a

una decisión de acotamiento, se dice que este nodo ha sido podado. Para que efectivamente el acotamiento sea efectivo, cotas inferiores (duales) y superiores (primales) adecuadas deben establecerse. Las cotas inferiores son obtenidas con las relajaciones lineales de los problemas Q_{LR} mientras que las cotas superiores son establecidas en el paso (10) del algoritmo, sin embargo también pueden encontrarse con las denominadas heurísticas primales.

El siguiente algoritmo describe el esquema de *Branch & Bound* de forma genérica:

Algorithm 2: Algoritmo de *Branch & Bound*

Input: Problema P de minimización.
Result: Solución óptima $x^* \in X_{MIP}$ y su valor objetivo $c^* = c^T x^*$, o la conclusión de que $X_{MIP} = \emptyset$ y $c^* := \infty$.

1 Inicializar $\mathbb{S} := \{P\}$ y $\overline{c} := \infty$.
2 **if** $\mathbb{S} = \emptyset$ **then**
3 | **return** $c^* = \overline{c}$ y x^* (si existe).
4 **end**
5 Escoger un problema $Q \in \mathbb{S}$ y fijar: $\mathbb{S} = \mathbb{S} \setminus \{Q\}$ Se resuelve la relajación lineal de Q, Q_{LR}. Si Q_{LR} es vacío entonces fijar $c^*_Q := \infty$. En cambio, si hay una solución óptima se obtiene x^*_Q y c^*_Q.
6 **if** $c^*_Q \geq \overline{c}$ **then**
7 | Volver al paso 2
8 **end**
9 **if** $x^*_Q \in X_{MIP}$ **then**
10 | Fijar $\overline{c} = c^*_Q$ y $x^* = x^*_Q$
11 **end**
12 **Branching**: Dividir el problema Q en subproblemas $Q = Q_1 \cup \ldots \cup Q_k$ y se fija: $\mathbb{S} = Q \cup \{Q_1, \ldots, Q_k\}$. Volver al paso 2.

La selección de nodos, realizada en el paso (5), y las decisiones de ramificación (*branching*), en el paso (12), toman decisiones determinantes en el esquema de *B&B* y, por tanto, deben diseñarse de forma de que hagan uso de la estructura del problema a resolver. Ambas decisiones tienen un profundo impacto en que tan temprano en la ejecución del esquema se encuentran buenas soluciones primales y qué tan rápido las cotas inferiores de los problemas abiertos en $\mathbb{S}$ aumenta.

En Achterberg et al. (2005) se detalla que, dado un problema Q, la única forma de particionarlo en subproblemas en el contexto de un esquema de *B&B* para un problema lineal es ramificar en desigualdades lineales, para mantener así las propiedades de un problema lineal relajado. La forma más común y fácil de hacer lo anterior es con *desigualdades triviales*, estas son desigualdades que separan los intervalos factibles para una variable en particular. Específicamente, si tenemos una variable i con un valor no entero en la solución relajada actual x^*_Q, vemos las diferencias respecto a sus valores enteros más cercanos $f_i^+ = \lceil x^*_{i,Q} \rceil - x^*_{i,Q}$ y $f_i^- = x^*_{i,Q} - \lfloor x^*_{i,Q} \rfloor$. El primer subproblema se obtiene agregando la restricción $x_i \leq \lfloor x^*_{i,Q} \rfloor$ (llamado subproblema izquierdo o hijo izquierdo y es denotado por Q_i^-) y el segundo al agregar la restricción alternativa $x_i \geq \lceil x^*_{i,Q} \rceil$ (llamado subproblema o hijo derecho, denotado por Q_i^+). Esta regla de ramificación es también llamada ramificación en variables (*branching on variables*) puesto que sólo requiere cambiar las cotas de la variable en cuestión, x_i.

El algoritmo (3), encontrado en Achterberg et al. (2005), muestra la forma de selección de la variable específica que va a ser objeto de *branching*. Los pasos (2) y (3) del algoritmo dependen de una función que evalúe las variables con valor decimal del subproblema en que se está trabajando, cada estrategia de *branching* tiene una idea diferente para dicha función. La calidad de una ramificación es medida en el cambio en la función objetivo de las relajaciones lineales de los subproblemas resultantes (Q_i^- y Q_i^+) con respecto del problema padre, Q.

Algorithm 3: Algoritmo genérico de selección de variable.

Input: Actual problema Q con una solución óptima $x_Q^* \notin X_{MIP}$.
Result: Indice i perteneciente a una variable no entera en solución x_Q^*.

1 Sea $C = \{i \in I | x_{i,Q}^* \notin \mathbb{Z}\}$
2 Para cada candidato $i \in C$, se calcula un valor $s_i \in \mathbb{R}$
3 **return** Indice $i \in C$ que cumple con: $s_i = \text{máx}_{j \in C}\{s_j\}$

La siguiente figura ejemplifica la búsqueda de soluciones en un esquema de *Branch & Bound*, el algoritmo (3) estaría siendo ejecutado en el nodo actual (marcado como Q) y generando nuevos subproblemas que ingresarían al set $\mathbb{S}$, para que en la siguiente iteración del esquema uno de los elementos de dicho set sea escogido en el paso (5) del algoritmo (2).

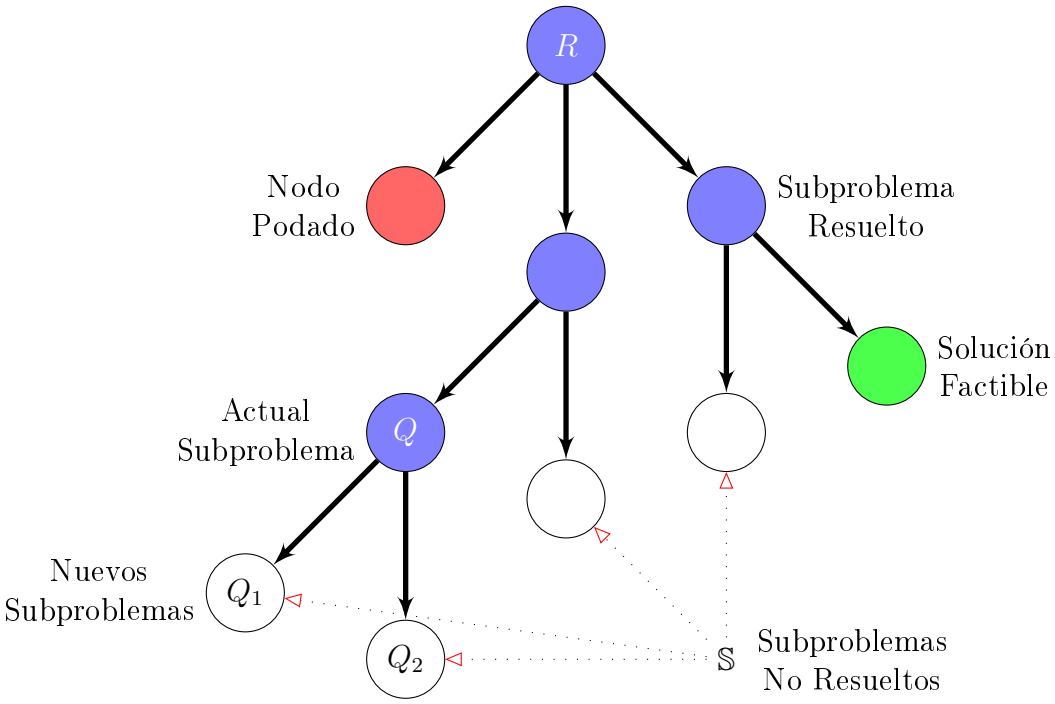


Figura 1.1: Árbol de ramificación en esquema *Branch & Bound*

Con el fin de comparar variables candidatas a ser ramificadas, los cambios que estas generan, al ser ramificadas, en la función objetivo del problema linealmente relajado deben ser obtenidos: $\delta_i^- := c_{Q_i^-}^* - c_Q^*$ y $\delta_i^+ := c_{Q_i^+}^* - c_Q^*$, o al menos es necesario obtener estimaciones de estos cambios. Estos valores han de ser sopesados en una función del tipo encontrada en Linderoth and Savelsbergh (1999):

$$g\left(q^-, q^+\right) = (1-\mu)\,\text{mín}\{q^-, q^+\} + \mu\,\text{máx}\{q^-, q^+\} \tag{1.28}$$

El valor μ es un número entre 0 y 1, usualmente determinado empíricamente. Puesto que la función (1.28) necesita un tratamiento especial en caso de que alguno de los subproblemas de Q (Q^- y Q^+) sea infactible, en Achterberg (2007) expone la siguiente función, que es utilizada por la librería SCIP:

$$g\left(q^-, q^+\right) = \text{máx}\{q^-, \varepsilon\} \cdot \text{máx}\{q^+, \varepsilon\} \tag{1.29}$$

En donde $\varepsilon = 10^{-6}$. En lo que sigue se describen las estrategias de ramificación y selección de nodos presentes en Achterberg et al. (2005), Achterberg (2007) y Morrison et al. (2016).

1.2.1.1. Estrategias de Ramificación

Una estrategia común y fácil de implementar es la estrategia de la variable **menos factible**, en dicha estrategia se escoge la variable más lejana a un valor entero, o sea, aquella cuya parte decimal sea la más cercana a 0.5: $s_i = \text{mín}\{x^*_{i,Q} - \lfloor x^*_{i,Q} \rfloor, \lceil x^*_{i,Q} \rceil - x^*_{i,Q}\}$. Los resultados obtenidos para esta estrategia en Achterberg et al. (2005) muestran que no es mejor que escoger variables de forma aleatoria, en términos del tiempo computacional requerido y el número de subproblemas resueltos. Menos común es la estrategia inversa, es decir, aquella que escoge la variable **más factible**, ergo, la más lejana a 0.5: $s_i = \text{máx}\{x^*_{i,Q} - \lfloor x^*_{i,Q} \rfloor, \lceil x^*_{i,Q} \rceil - x^*_{i,Q}\}$. Tal como la estrategia anterior, los resultados entregados por esta estrategia son bastante pobres (ver Achterberg et al. (2005)).

Una estrategia más sofisticada es ***pseudocost branching***, propuesta en Bénichou et al. (1971), intenta predecir el cambio en la función objetivo por decimal de distancia al próximo entero. Para ver esto primero se define ζ_i^+ como la ganancia en la función objetivo por unidad de cambio en la variable i en el subproblema Q:

$$\zeta_i^- = \frac{\delta_i^-}{f_i^-} \qquad \text{y} \qquad \zeta_i^+ = \frac{\delta_i^+}{f_i^+} \tag{1.30}$$

Donde f_i^+ es la distancia del valor de la variable decimal i en el subproblema Q al siguiente entero (análogamente se puede inferir f_i^-), como se explico anteriormente. Sea σ_i^+ la sumatoria de los ζ_i^+ sobre todos los problemas Q donde la variable i haya sido inicializada y cuya relajación Q_i^+ haya sido resuelta y resultara factible. Se enumeran dichos problemas en la variable η_i^+ y se definen los pseudocostos para la variable i como:

$$\Psi_i^- = \frac{\sigma_i^-}{\eta_i^-} \qquad \text{y} \qquad \Psi_i^+ = \frac{\sigma_i^+}{\eta_i^+} \tag{1.31}$$

Usando la función (1.28) se obtiene el valor de: $s_i = g\left(f_i^+\Psi_i^+, f_i^-\Psi_i^-\right)$, y se calculan los valores necesarios para ejecutar el algoritmo (3) utilizando *pseudocost branching*.

El evidente problema de esta estrategia de ramificación es que al inicio tendremos $\sigma_i^+ = \sigma_i^- = \eta_i^+ = \eta_i^- = 0$ para toda variable del problema. Para los valores de Ψ_i^+ y Ψ_i^-, estos tomarán el promedio de las variables inicializadas en dicha dirección cuando la variable i no lo este, es decir, $\Psi_i^+ = \Psi_{AVG}^+$. En el caso de que ninguna variable este inicializada en dicha dirección el valor será fijado en 1 (homólogo para el otro caso).

Otra estrategia muy común y disponible en varios software de optimización es ***strong branching***, desarrollada en el contexto del problema del vendedor viajero en Applegate et al. (1995) se ha vuelto parte estándar de varias librerías como CPLEX. Lo que trata de hacer *strong branching* es probar cuál variable es la que entrega un mayor avance en la cota inferior antes de efectivamente ramificar en alguna de ellas. La prueba es realizada introduciendo, transitoriamente, para cada variable primero una cota superior $x_i \leq \lfloor x_{i,Q}^* \rfloor$ y, posteriormente, una inferior $x_i \geq \lceil x_{i,Q}^* \rceil$.

Dentro de esta estrategia, cuando se toma todo el set de variables no enteras en la solución lineal del nodo como candidatos y sus LP se resuelven a optimalidad, entonces estamos hablando de ***full strong branching***. En *full strong branching* básicamente se trata de encontrar la variable que localmente presenta la más grande mejora en la función objetivo del problema linealmente relajado (o función particular que se utilice para evaluar ramificaciones). Los resultados mostrados en Achterberg et al. (2005) indican que elegir la mejor opción local obtiene un mejor resultado en comparación a otras estrategias respecto del número de nodos que deben resolverse. Sin embargo el costo computacional por nodo es alto y se hace impráctico su uso en grandes problemas, en consecuencia gran parte de las estrategias de ramificación tratan de obtener un estimado de lo que *strong branching* obtiene en cada nodo.

Para acelerar *strong branching* se han pensado dos avenidas. La primera es restringir el set de variables decimales candidatas a ser ramificadas en conjunto de variables de interés. La segunda es limitar el número de iteraciones de del algoritmo de simplex dual que se realizarán en cada evaluación, el argumento detrás de la idea es que el cambio en la función objetivo decrece a medida que se hacen más iteraciones. Esto nos entrega dos parámetros a especificar en *strong branching*: El tamaño del set de candidatos a evaluar (κ) y el número de iteraciones de simplex dual que se realizan en cada una de las evaluaciones (γ).

Inclusive implementando las medidas recién expuestas para acelerar *strong branching*, este suele consumir considerables recursos computacionales. Adicionalmente, a mayor esfuerzo por acelerar el proceso de dicha estrategia, menos precisa será. Si recordamos, la debilidad de *pseudocost branching* es su inicio, en donde no hay información para obtener pseudocostos y los valores s_i sólo van a reflejar la fraccionalidad de las variables. Desafortunadamente gran parte de los nodos iniciales (de cada variable) del árbol de ramificación se encuentran cercanos al nodo raíz (parte superior del árbol) donde las decisiones de ramificación tienen mayor impacto en la estructura del árbol y de los subproblemas (Achterberg (2007)), en consecuencia las decisiones tomadas por *pseudocost branching* al inicio del esquema no son tomadas con información confiable.

Para atacar dichos problemas y sacar lo mejor de cada estrategia se unen en ***hybrid***

strong/pseudocost branching, en donde *strong branching* es utilizado en la parte superior del árbol hasta una cierta profundidad d. Para los nodos cuya profundidad sea mayor a d se utilizará *pseudocost branching*.

A pensar de los esfuerzos de la estrategia anteriormente descrita, persiste un problema con respecto a la parte inferior de árbol de ramificación y es que pasada la profundidad d pueden haber variables todavía no inicializadas, en cuyo caso el pseudocosto no será confiable (Linderoth and Savelsbergh (1999)). Para mitigar tal riesgo se propone emplear *strong branching* en toda variable no inicializada en vez en los casos cuando un nodo se encuentre a un profundidad menor a d. Este esquema es llamado ***pseudocost branching with strong initialization***.

Las últimas dos estrategias son generalizadas en ***reliability branching*** (Achterberg et al. (2005)), de forma que *strong branching* es utilizado no sólo en variables no inicializadas sino también en aquellas en donde su pseudocosto no sea confiable. Conforme a lo anterior el pseudocosto de una variable i no será confiable cuando se cumpla la condición: $\text{mín}\{\eta_i^-, \eta_i^+\} < \eta_{rel}$, donde η_{rel} es un parámetro de confiabilidad. En esta regla de ramificación al iterar se separan aquellas variables con pseudocostos variables de los que no, sobre las variables que no cumplen la condición de confiabilidad se realizan un número γ de iteraciones de simplex dual y se obtienen los valores de sus subproblemas izquierdo y derecho, y las mejoras locales en función objetivo. Con las mejoras locales los pseudocostos de dichas variables se actualizan y se agregan al set pseudocostos confiables para su comparación.

1.2.1.2. Selección de Nodos

Posterior a la resolución y procesamiento de un subproblema, el esquema de *B&B* escoge otro subproblema de entre el set de subproblemas que todavía no son procesados. La elección del subproblema a resolverse a continuación usualmente tiene como fin dos metas que suelen ir en direcciones opuestas:

i. Encontrar buenas soluciones factibles al problema original con el fin de mejorar la cota superior, lo que ayuda a acotar la búsqueda del esquema de *B&B* en el árbol de ramificación.

ii. Mejorar la cota inferior del problema.

Respecto al primer fin de la selección de nodos tenemos ***Depth First Search***, regla inicialmente utilizada en Little et al. (1963) para el problema del vendedor viajero y en Dakin (1965) se generalizó para problemas de programación entera. Esta regla de selección de nodos apunta a consistentemente escoger un nodo hijo del actual como el siguiente a ser resuelto. Si el actual nodo es podado, la regla indica de debe retrotraerse hasta el nodo con hijos no resueltos más reciente y allí escoger alguno de ellos.

Esta regla es usualmente empleada en problemas que buscan soluciones factibles mas que óptimas, como es el caso de los problemas de satisfacción de restricciones, CSP y SAT. Sin embargo, la idea principal de *depth first search* lleva inherentemente a otro beneficio asociados a su aplicación, y es que el próximo subproblema (nodo hijo) a resolver es muy similar al

actual, en consecuencia los cambios son mínimos y los recursos necesarios para llevar a cabo la regla no son muy altos. De hecho su tercera ventaja respecto a otras reglas de selección de nodos es su bajo consumo de memoria. Si la profundidad del nodo siendo procesado es d, entonces el arbol de ramificación se contiene a lo más $d \cdot b + 1$ nodos sin podar, donde b es el número máximo de hijos que un nodo puede tener (Achterberg (2007)). Lo anterior implica que para problemas de con ramificaciones con 2 opciones (*2-way branching*, variables binarias por ejemplo) los nodos a almacenar no excede el valor $2d_{max} + 1$, donde d_{max} es la profundidad máxima del árbol de ramificación.

En contrapartida a *depth first search*, ***Best First Search*** va en pos del segundo fin de la selección de nodos, mejorar la cota inferior global de problema lo más rápido posible. Esta regla apunta hacia esa meta siempre escogiendo el subproblema con la cota inferior más pequeña de entre los nodos remanentes en el árbol. Como beneficio secundario de esta regla es que conlleva que el número de nodos que deben se procesados es mínimo, dada una regla de ramificación fija (prueba de esto se puede encontrar en Achterberg (2007)). Sin embargo, no hay una forma única de ejecutar esta regla puesto múltiplos nodos pueden presentar el mismo valor y hay se debe definir una estrategia para resolver dichas situaciones.

Como ya fue explicado *best first search* lleva a una menor cantidad de nodos procesados mientras que *depth first search* obtiene soluciones factibles rápidamente y suele tener procesos de solución de nodo más rápidos debido a la similaridad de subproblemas consecutivos. En ***Best First with Plunging*** se trata de obtener lo mejor de ambas estrategias de búsqueda con la idea de *plunging*. Esta idea indica que se deben procesar los nodos hijos o hermanos del actual nodo antes de volver a iterar bajo la lógica de *best first search*. La mayor desventaja de *depth first search* es el riesgo de procesar nodos que hubiesen sido podados de haber tenido una mejor cota inferior. *Plunging* (profundización) posee el mismo problema, pero en un grado menor puesto que al final de la profundización la elección del siguiente nodo se realiza respecto a la mejor cota inferior entre los nodos remanentes. Durante cada profundización se realizan un cierto número de iteraciones de *depth first search*, pero se finaliza si es que se supera el umbral de pasos o si el gap relativo supera el umbral γ_{max}:

$$\gamma\left(Q\right) = \frac{\widehat{c}_Q - \underline{\widehat{c}}}{\widehat{c} - \underline{\widehat{c}}} \tag{1.32}$$

Donde $\widehat{c}_Q$ es la cota inferior del actual subproblema, $\underline{\widehat{c}}$ es la cota inferior global del problema y $\widehat{c}$ es la cota superior global. Por otro lado los números mínimos y máximos de profundizaciones son valores dinámicos que usualmente se obtienen como $0{,}1 \cdot d_{max}$ y $0{,}5 \cdot d_{max}$ respectivamente, en donde d_{max} es la máxima profundidad de todos los nodos procesados (Achterberg, 2007). Lo anterior implica que al comienzo del algoritmo de $B\&B$ casi no se hacen profundizaciones.

Como se ha dicho anteriormente, *best first search* obtiene buenas cotas inferiores rápidamente pero sus soluciones suelen estar lejos de los requerimientos de integralidad mientras que *depth first search* rápidamente abandona la zona del árbol de ramificación en donde encontramos buenos valores objetivos hacia donde se puedan encontrar soluciones factibles de manera expedita. En consecuencia ninguna de ambas trata de buscar soluciones de buen valor objetivo pero que no se alejen demasiado de ser factibles. ***Best Estimate Search*** apunta a obtener soluciones de este tipo estimando el valor e_Q para la posible solución factible que

se encuentra en los sub-árboles que representan los descendiente del nodo Q. La regla luego escoge el nodo que minimiza este estimado. En el cálculo de e_Q se involucran tanto las cotas actuales del problema como información del nivel de integralidad de la solución lineal.

En Achterberg (2007) se mencionan dos esquemas para el cálculo de e_Q. El primero se basa en la idea de *best projection* de Bénichou et al. (1971), que calcula el estimado de la siguiente forma:

$$e_Q^{proj} = \widehat{c}_Q + \left(\frac{\widehat{c} - \widehat{c}_R}{\phi\left(\widehat{x}_R\right)}\right) \phi\left(\widehat{x}_Q\right) \tag{1.33}$$

Donde $\widehat{x}_Q$ es la solución al actual subproblema, $\widehat{x}_R$ es la solución del problema en el nodo raíz y $\phi\left(\widehat{x}\right)$ es una medida de la fraccionalidad del vector $\widehat{x} \in \mathbb{R}^n$ y se define como:

$$\phi\left(\widehat{x}\right) = \sum_{j=1}^{n} \phi\left(\widehat{x}_j\right) \quad \text{donde} \quad \phi\left(\widehat{x}_j\right) = \text{mín}\{\widehat{x}_j - \lfloor \widehat{x}_j \rfloor, \lceil \widehat{x}_j \rceil - \widehat{x}_j\} \tag{1.34}$$

La interpretación de *best projection* indica que uno puede calcular una estimación del cambio en la función objetivo del actual subproblema vía el cambio por unidad de decrecimiento de la fraccionalidad de la solución en el nodo raíz y se asume que la actual solución puede ser conducida a factibilidad vía el mismo incremento por unidad de fraccionalidad.

El segundo modo de cálculo de la estimación es mediante el uso de pseudocostos, este esquema se puede reastrear hasta Forrest et al. (1974), y el valor de la función objetivo se calcula como:

$$e_Q = \widehat{c}_Q + \sum_{j \in I} \text{mín}\{f_i^+ \Psi_i^+, f_i^- \Psi_i^-\} \tag{1.35}$$

En el trabajo de Linderoth and Savelsbergh (1999) se hacen exhaustivas comparaciones en donde se establece que, de haber pseudcostos confiables, la regla de Forrest et al. (1974) es superior a la de *best projection*.

Al igual que con *best first search*, *best estimate search* puede ser combinado con *depth first search* al incluir el concepto de *plunging* en su implementación, nuevamente tratando transferir el beneficio de tener un aumento en el procesamiento de nodos debido a la semejanza entre subproblemas sucesivos a *best estimate search*.

Una regla adicional que trata de entrelazar y aprovechar las virtudes de las 3 reglas anteriores es ***Hybrid Best Estimate/Best First Search***. Dicha regla le da mayor importancia a la cota inferior global del problema fijado la regla de selección de nodos como una suma ponderada entre los valores de *best estimate* y *best first*. En esta regla el valor escogido es aquél que minimiza dicha suma ponderada:

$$e_{Hybrid} = w \cdot e_Q + (1 - w) \cdot \widehat{c}_Q \tag{1.36}$$

Con $w \in [0, 1]$. Esta estrategia es usualmente utilizada en conjunto con *plunging*, entregando a esta regla parte de las propiedades favorables de las reglas hasta ahora cubiertas.

Como contrapartida a *depth first search* esta la regla ***Breath First Search***, cual apunta a explorar todos los subproblemas existentes a una cierta distancia del nodo raíz antes de procesar cualquier otro nodo a una mayor profundidad. La ventaja de esta estrategia es que rápidamente encuentra soluciones que se encuentran cerca de la raíz del árbol de ramificación, operando de esta forma bien en árboles desbalanceados. Sin embargo, como las soluciones óptimas completas usualmente se encuentran a profundidades largas, esta regla generalmente se ve dificultada de explotar reglas de podado que comparan respecto a la actual solución incumbente. En consecuencia la cantidad de memoria que usa es usualmente alta, lo que la hace una regla poco popular en el contexto de $B\&B$.

En otro frente, la regla ***Cyclic Best First Search***, inicialmente llamada *distributed best first search* en Kao et al. (2009), se ha vuelto popular en aplicaciones en un buen número de problemas de *schedueling*. Esta regla puede ser interpretada como otro híbrido entre *best first search* y *depth first search*. Si bien usualmente *best first* es aplicada utilizando una sola estructura de almacenamiento para los nodos no explorados, en esta nueva regla los nodos no explorados son almacenados en un conjunto de estructuras que los autores llaman *contours*. Cuando un nuevo subproblema es creado, este es asignado a alguno de los *contours* según alguna regla específica (ejemplo puede ser que cada *contour* esta asociado a una profundidad específica del árbol). Tal como su nombre los sugiere, al explorar el espacio de búsqueda esta estrategia itera entre los diferentes *contours* seleccionando en cada uno de ellos el mejor subproblema (según el criterio usado en *best first search*).

Al separar los subproblemas en *contours* la estrategia puede asignar un ranking según la medida utilizada en *best first search* en cada *contour*, en vez de hacerlo globalmente. El beneficio de lo anterior es que se pueden hacer comparaciones entre subproblemas más similares, asignando en un mismo *contour* subproblemas comparables entre sí. La rotación entre los *contours* hace que diversos frentes sea explorados de forma sistemática, diversificando la búsqueda por nuevas soluciones incumbentes.

1.2.2. Branch & Price

La resolución exacta de problemas de programación lineal mixta de gran tamaño descansa en gran medida en que la relajación lineal de los problemas formulados sean buenas aproximaciones de casco convexo de las soluciones factibles. En la literatura, particular atención ha tenido el algoritmo de *Branch & Cut*, cuya idea básica es relativamente simple: clases de desigualdades válidas (*valid inequalities* en inglés) son excluidas del problema relajado bajo el supuesto de que en la solución óptima la mayor parte de ellas no serán activas. Luego, si al encontrar una solución al problema relajado esta resulta ser infactible, un subproblema

es resuelto en búsqueda de restricciones violadas dentro de cada clase de desigualdades válidas. Las restricciones violadas son incorporadas en el problema relajado para excluir dicha solución de la región factible, luego el problema relajado es re-optimizado. La ramificación y acotamiento ocurre cuando no se pueden encontrar más desigualdades válidas violadas. En resumen *Branch & Cut* es una generalización de *B&B* que permite la generación de cortes a lo largo del árbol de ramificación (Mitchell, 2009).

Análogamente, en la resolución de grandes *MILP*, está la idea de considerar formulaciones con una gran número de variables, tal como sugieren Barnhart et al. (1998):

i. Una formulación compacta para un *MIP* puede tener una relajación lineal débil. Esta relajación puede ser mejorada con una formulación que contiene una enorme cantidad de variables.

ii. Una formulación compacta de un *MIP* puede tener una estructura simétrica que lleva al algoritmo de *B&B* a tener un pobre rendimiento ya que el problema apenas cambia después de la ramificación (imaginemos la ramificación en un problema de ruteo sobre una variable de arco). La reformulación del problema en uno con gran cantidad de variables puede eliminar dicha simetría.

iii. Generación de columnas provee una descomposición del problema en maestro y subproblemas, que puede tener una interpretación natural en el contexto del problema que permite la fácil adición de restricciones.

iv. Una formulación con un gran número de variables podría ser la única forma de abordar el problema computacionalmente.

La idea central de ***Branch & Price*** (*B&P*) es la integración del algoritmo de generación de columnas con el esquema de *Branch & Bound* con el mismo fin de *B&C*: la resolución de problemas lineales mixtos de gran tamaño. Como generación de cortes, en *B&P* la generación de columnas se utiliza como herramienta para estrechar las relajaciones lineales del problema entero. En Barnhart et al. (1998) se explica que, al igual que en generación de columnas, en *B&P* se dejan fuera del problema relajado un conjunto importante de columnas puesto que son demasiadas para manejar de forma efectiva y, en la solución óptima al problema, gran parte de sus variables asociadas estarán fijas en cero. Luego, para evaluar optimalidad de la solución al problema relajado, se llama al subproblema de generación de columnas (o problema de *pricing*, similar a un problema de generación de cortes en el dual del problema relajado) el que es resuelto en búsqueda de columnas que puedan ingresar a la base del problema, para posteriormente re-optimizarlo. Por el contrario, de no encontrarse ninguna columna que pueda beneficiar al problema relajado, y si la solución encontrada no satisface las restricciones de integralidad, se procede a la ramificación. Es decir, *B&P* aplica generación de columnas en cada nodo del árbol de *B&B* (Savelsbergh, 2009).

Combinar los algoritmos de generación de columnas y *B&B* no es trivial y conlleva una serie de dificultades no menores, tal como Appelgren (1969) y Johnson (1989) demuestran en sus desarrollos iniciales. Entre las dificultades fundamentales de aplicar generación de columnas en conjunto con *B&B* a problemas enteros se mencionan:

i. Convencionalmente en programación entera se utiliza la ramificación en variables, lo cual puede impactar el subproblema de *pricing* al fijar variables en valores específicos.

ii. Resolver los subproblemas de generación de columnas a optimalidad a lo largo del árbol de *Branch & Bound* puede no ser eficiente, en tal caso diferentes reglas deberán generarse para manejar el árbol de ramificación resultante del esquema de *B&P* (Johnson, 1989). Lo que hace que la convergencia del problema sea lenta, e inclusive llegar a ser computacionalmente prohibitiva.

Prueba de lo anterior es que los primeros desarrollos del algoritmo fueron específicos al problema enfrentado en cada trabajo, como se puede observar en Vance et al. (1994) para un *bin packing problem* y Mehrotra and Trick (1996) para el *vertex coloring problem*. En la segunda mitad de los noventa, los trabajos de Vanderbeck and Wolsey (1996) y Barnhart et al. (1998) son los primeros estudios que no están atados a un problema específico y tratan de entregar un mapa general para la aplicación de *Branch & Price*. Actualmente *B&P* es una técnica bien establecida de resolución de problemas enteros de gran tamaño y de alta dificultad. El fuerte de *B&P* esta en el uso de la descomposición del problema de forma de explotar estructuras conocidas dentro de él y la posibilidad de resolver los subproblemas de generación de columnas eficientemente con algoritmos específicos a cada problema (*elementary shortest path problem with resource constraints* resuelto vía programación dinámica para problemas de ruteo, por ejemplo). Sin embargo, estas características que la distinguen también son las que hacen que no sea un método fácil de implementar puesto que los dos componentes más importantes, la rutina de generación de columnas y el esquema de *branching*, deben de ser implementados por el usuario.

Adicionalmente, en Barnhart et al. (1998), se describen los tipos de problemas para los cuales generación de columnas (y en consecuencia *B&P*) puede entregar una ventaja en su resolución. Los autores destacan los problemas tipo *set covering*, *set partitioning* y aquellos problemas en donde una gran cantidad de columnas es la única opción, puesto que su descomposición no es posible. De lo anterior es posible ver que los problemas de ruteo pueden ser reformulados como problemas *set covering* o *set partitioning*, de modo que las columnas del problemas representan rutas factibles, dejando que sea el set de restricciones el que haga la diferencia entre el tipo de problema en que se reformuló el problema de ruteo. Si cada cliente debe ser visitado exactamente una vez entonces estamos hablando de una formulación *set partitioning* mientras que si la restricción sobre las visitas a cada cliente indican que estos deben ser visitado al menos una vez, entonces hablamos de una formulación tipo *set covering*.

En Feillet (2010) podemos encontrar una profundización de la aplicación de generación de columnas y *B&P* para problemas de ruteo de vehículos, utilizando la variante con ventanas de tiempo como ejemplo. El problema es reformulado como un *set covering problem* y el autor indica que el problema no puede ser abordado únicamente por el algoritmo de *B&B* puesto que la cantidad de rutas factibles crece exponencialmente con el número de clientes (nodos) en el problema y, en consecuencia, es necesario aplicar generación de columnas, que transforma el método de resolución en un *Branch & Price*. El autor hace referencia a los diferentes aspectos en los que se debe poner atención al implementar el algoritmo. Dentro de esos aspectos se encuentran el esquema de ramificación, la implementación del subproblema de generación de columnas y su eficiencia, el set inicial de columnas que se le entrega al algoritmo y la

necesidad de implementar una solución al potencial problema de infactibilidad de alguno de los problemas maestros restringidos. Adicionalmente, en Barnhart et al. (1998), indican que la elección del set inicial de variables no es trivial, puesto que estas determinan las variables duales que son entregadas al subproblema, de modo que dicho set puede ser importante en la resolución del problema. Otro punto levantado es la necesidad de que al escoger un esquema de ramificación, este no entorpezca ni destruya el subproblema de generación de columnas.

1.2.3. Column Dependent Rows

En la literatura, el problema de *column-dependent-rows* (CDR en adelante) es un dilema que suele verse sólo en formulaciones de problemas de programación lineal de gran tamaño, con un número exponencial de variables. La propiedad clave de este tipo de problemas es la existencia de un set de restricciones que se entrelazan con las variables de interés. Dicho set tiene la dificultad de que es o demasiado grande para ser incorporado en el problema directamente o sus elementos sólo se pueden identificar en la medida que las variables que los definen son generadas de forma explícita.

Justamente uno de los supuestos claves en generación de columnas es que el set de restricciones estructurales en el problema maestro es conocido desde un inicio y se encuentra fijo, de modo que toda la información dual es entregada al subproblema. En problemas CDR la falta de variables duales, consecuencia de las restricciones faltantes en el maestro, imposibilita el correcto cálculo de los costos reducidos de las variables ausentes del maestro.

En consecuencia, la literatura de problemas CDR trata de resolver el problema de generar columnas y las restricciones que se generan de su creación estimando de manera correcta los valores de las variables duales faltantes. Dicha literatura es algo limitada y de hecho uno de los primeros algoritmos diseñados para su resolución fue planteado por Zak (2002) y es específico para la resolución de un *multistage cutting stock problem*. Su algoritmo consistía de tres partes. Los primeros trabajos que proponen esquemas genéricos para problemas CDR son los de Feillet et al. (2010) y Muter et al. (2013).

Es importante resaltar el hecho de que los problemas CDR se basan en la creación simultánea de filas y columnas, lo que es fundamentalmente diferente a lo realizado en el algoritmo de *Branch & Cut & Price*, puesto que este último algoritmo busca fortalecer la relajación lineal del problema vía la inclusión de desigualdades válidas en el árbol de búsqueda antes de hacer generación de columnas.

En la literatura encontramos dos tipos de solución a los problemas CDR, en el primero podemos encontrar el trabajo de Muter et al. (2013), en donde se especifica la forma canónica de un modelo genérico que representa el tipo de problemas CDR que, bajo ciertos supuestos, se pueden resolver con su método. Por otro lado esta el trabajo de Feillet et al. (2010), en donde se describe un procedimiento genérico para la solución de este tipo de problemas que depende de la construcción de soluciones duales que permitan el cálculo de los costos reducidos de forma adecuada, sin embargo no describen una regla específica sobre el cómo construir dicha solución dual e indican que esto se debe a que cada aplicación debe generar una forma particular de construcción de dicha solución tomando en cuenta las particularidades de la

aplicación que se esté resolviendo.

En la formulación genérica propuesta por Muter et al. (2013) de problemas CDR, el maestro, en adelante llamado GMP (*Generic Master Problem*), es presentado en las expresiones (1.37) a (1.43). En este problema genérico existen un número exponencial de variables y_k y x_n, y se permite la incorporación de ambas vía generación de columnas al resolver el GMP. Se asume que las restricciones (1.39) y (1.40) son conocidas de antemano en su totalidad, mientras que el set de restricciones del tipo (1.41) no está explícito, dado que estas son la restricciones que entrelazan las variables x_n e y_k y para obtener el set completo necesitaríamos conocer el set completo de tales variables, e incluso en tal caso estas podrían ser demasiadas para ser incorporadas directamente en el problema.

El problema restringido del GMP, llamado SRMP (*Short Restricted Master Problem*) por los autores, es prácticamente el mismo sólo que se utilizan los set $\bar{K} \subset K$, $\bar{N} \subset N$ para representar los sets de variables y_k y x_n presentes en el problema. Adicionalmente se utiliza el set $I(\bar{K}, \bar{N}) \subset I$ para indicar el conjunto de restricciones de entrelazamiento presentes en el SRMP, formadas por las variables $\{y_k | k \in \bar{K}\}$ y $\{x_n | n \in \bar{N}\}$.

$$\text{mín} \quad \sum_{k \in K} c_k y_k + \sum_{n \in N} d_n x_n \tag{1.37}$$

$$\text{s.a} \tag{1.38}$$

$$\sum_{k \in K} A_{jk} y_k \geq a_j \quad \forall j \in J \tag{1.39}$$

$$\sum_{n \in N} B_{mn} x_n \geq b_m \quad \forall m \in M \tag{1.40}$$

$$\sum_{k \in K} C_{ik} y_k + \sum_{n \in N} D_{in} x_n \geq r_i \quad \forall i \in I \tag{1.41}$$

$$y_k \geq 0 \quad \forall k \in K \tag{1.42}$$

$$x_n \geq 0 \quad \forall n \in N \tag{1.43}$$

Durante la fase de generación de columnas de este problema nuevas variables $\{y_k | k \in S_K\}$ y $\{x_n | n \in S_N\}$, donde $S_K \subseteq \left(K \setminus \bar{K}\right)$ y $S_N \subseteq \left(N \setminus \bar{N}\right)$, son agregadas al SRMP producto de la resolución de diferentes subproblemas. Adicionalmente estas nuevas variables puede que aparezcan en nuevas restricciones tipo (1.41) que en dicho momento no se encuentran en el SRMP. El set que representa tales restricciones de entrelazamiento, ausentes en el maestro, se puede representar como $\Delta\left(S_K, S_N\right) = I\left(\bar{K} \cup S_K, \bar{N} \cup S_N\right) \setminus I\left(\bar{K}, \bar{N}\right)$. Es así como el SRMP crece tanto vertical como horizontalmente durante la generación de columnas. Los autores indican que tres supuestos caracterizan los tipos de problemas CDR que pueden ser resueltos por la metodología que proponen.

El primer supuesto indica que la generación de variables x depende de la generación de variables y, y de hecho, cada variable x esta asociada con sólo un set de restricciones de entrelazamiento.

Supuesto 1.2.1 La generación de un nuevo set (o conjunto) de variables $\{y_k | k \in S_K\}$

promueve la generación de un nuevo set de variables $\{x_n | n \in S_N(S_K)\}$. De hecho, una variable $x_{n'}, n' \in S_N(S_K)$ no aparece en ninguna otra restricción de entrelazamiento que no sea en las del set $\Delta(S_K, S_N(S_K))$, introducidas al SRMP en conjunto con $\{y_k | k \in S_K\}$ y $\{x_n | n \in S_N(S_K)\}$.

El set $S_N(S_K)$ es aquél que contiene las variables x_n generadas a partir de la incorporación del set de variables y_k S_K. Esto muestra la dependencia del set $\bar{N}$ al set $\bar{K}$.

El segundo supuesto requiere la definición del término *set de variables mínimo*. Un set de variables mínimo es aquel set de variables y que gatilla la generación de un set de variables x y las restricciones de entrelazamiento asociadas, tal como lo indica el supuesto 1.2.1.Con esta definición, el segundo supuesto indica que ninguna restricción de entrelazamiento es violada mientras no se hayan incorporado al SRMP todas las variables de al menos un set de variables mínimo asociado a ella.

Supuesto 1.2.2 Una restricción de entrelazamiento es redundante hasta que todas las variables en al menos uno de los sets de variables mínimos asociados a la restricción estén incluidos en el SRMP.

Con los supuestos 1.2.1 y 1.2.2 se delinea lo que finalmente será el esquema de subproblemas para la generación de columnas y filas propuesto en Muter et al. (2013). En dicho esquema tiene como objetivo identificar uno o varios set de variables mínimos, en donde cada uno de estos sets $\{y_k | k \in S_K\}$ propone la generación del set de variables $\{x_n | n \in S_N(S_K)\}$. Ambos sets aparecen en un set de restricciones de entrelazamiento $\Delta(S_K, S_N(S_K))$, actualmente ausentes en el SRMP.

El tercer supuesto caracteriza el signo de los coeficientes en las restricciones de entrelazamiento.

Supuesto 1.2.3 Supongamos que tenemos un set de variables mínimo $\{y_k | k \in S_K\}$ que genera un set de restricciones de entrelazamiento $\Delta(S_K, S_N(S_K))$ y un set de variables $\{x_n | n \in S_N(S_K)\}$. Cuando el set de restricciones de entrelazamiento $\Delta(S_K, S_N(S_K))$ es introducido en el SRMP durante la generación de columnas (y filas), para cada $k \in S_K$ existe una restricción $i \in \Delta(S_K, S_N(S_K))$ de la forma:

$$C_{ik} y_k + \sum_{n \in S_N(S_K)} D_{in} x_n \geq 0 \tag{1.44}$$

Donde $C_{ik} > 0$ y $D_{in} < 0$ para todo $n \in S_N(S_K)$.

El supuesto 1.2.3 asegura que una variable $x_n, n \in S_N(S_K)$ no puede asumir un valor positivo hasta que todas las variables en al menos un set de variables mínimas que genera $\Delta(S_K, S_N(S_K))$ son positivas en el SRMP.

Adicionalmente los autores hacen una segunda definición tangente a los sets de variables mínimas, los problemas CDR con y sin interacción. Aquellos con interacción son aquellos en donde la cardinalidad de los sets de variables mínimas es mayor a uno, mientras que en los problemas sin interacción la cardinalidad de dichos sets es exactamente uno.

Para clarificar hacia adelante se expone el dual del maestro presentado anteriormente:

$$\text{máx} \quad \sum_{j\in J} a_j u_j + \sum_{m\in M} b_m v_m + \sum_{i\in I} r_i w_i \tag{1.45}$$

$$\text{s.a} \tag{1.46}$$

$$\sum_{A_{jk}u_j} + \sum_{i\in I} C_{ik} w_i \leq c_k \qquad \forall k \in K \tag{1.47}$$

$$\sum_{m\in M} B_{mn} v_m + \sum_{i\in I} D_{in} w_i \leq d_n \qquad \forall n \in N \tag{1.48}$$

$$u_j \geq 0 \quad \forall j \in J, v_m \quad \forall m \in M, w_i \quad \forall i \in I \tag{1.49}$$

Donde u, v y w son las variables duales de las restricciones (1.39), (1.40) y (1.41), respectivamente. Se puede intuir que es la variable dual w la que genera el problema de CDR puesto que no todas las restricciones (1.41) están explícitas en el SRMP al comienzo de la resolución del problema.

El esquema de Muter et al. (2013) para problemas CDR implica el uso de 3 subproblemas, dos de generación de columnas y un tercero de generación de filas (de entrelazamiento). El primer subproblema es el clásico de generación de columnas, busca encontrar una variable $y_k, k \in \left(K \setminus \bar{K}\right)$ con costo reducido negativo. En este primer subproblema sólo se toman en cuenta las variables duales $w_i, i \in I\left(\bar{K}, \bar{N}\right)$. Al encontrar una variable y de costo reducido negativo es posible que algún set de variables mínimos se complete y produzca la incorporación de la restricción de entrelazamiento correspondiente, que a su vez generará la incorporación de un set de variables x.

El segundo subproblema trata de encontrar variables x al tratar de buscar restricciones tipo (1.48) violadas de forma de agregar las correspondientes columnas al SRMP. Sin embargo la incorporación de estas variables no genera la incorporación de restricciones de entrelazamiento debido al supuesto 1.2.2.

Finalmente, el tercer subproblema aborda el problema de las duales faltantes en un esquema de dos pasos. Este subproblema sólo se debe llevar a cabo cuando los dos anteriores no encuentran variables de costo reducido negativo, de forma consecutiva. En este subproblema se tratan de encontrar variables y que tendrían costo reducido negativo si es que un nueva restricción de entrelazamiento es introducida al SRMP. En ese sentido la distinción entre problemas con y sin interacción se hace relevante puesto que problemas sin interacción sólo necesitan identificar una variable para inducir la creación una o más restricciones de entrelazamiento, sin embargo, para los problemas con interacción el subproblema debe identificar uno o varios sets de variables mínimas para hacer que una variable y tenga su costo reducido negativo (gracias a la introducción de la restricción de entrelazamiento asociada al set de variables mínimas identificado).

En vista de lo anteriormente explicado, en la primera parte se obtiene los valores de las variables w para el set de restricciones generadas al agregar un set de variables mínimas asociadas a una variable y específica, lo cual se repite para todos los sets mínimos de la familia de de sets mínimos asociados a dicha variable, que vendría a representar el valor óptimo de $\sum_{i\notin I\left(\bar{K},\bar{N}\right)} C_{ik} w_i$. Posteriormente, en la segunda parte, se escoge la variable y de menor costo

reducido dada la familia de sets de variables mínimas de mayor valor. Implícitamente lo que hace este subproblema es construir una solución básica para el SRMP que incluye el set de restricciones de entrelazamiento que permiten que una variable y obtenga un costo reducido negativo. Esto es muy parecido al esquema planteado por Feillet et al. (2010), explicado más adelante. De hecho, una condición necesaria, y probada, es que los valores duales de $\{v_m | m \in M\}$, $\{u_j | j \in J\}$ y $\{w_i | i \in I\left(\bar{K}, \bar{N}\right)\}$ permanecen sin cambio en la solución construidas y siguen siendo óptimas, similar a lo propuesto en Feillet et al. (2010).

En la figura 1.2 se observa el esquema completo propuesto de Muter et al. (2013), en donde la variable **FLAG** se cambia a 1 cada vez que alguno de los tres subproblemas encuentra una variable de costo reducido negativo que entra al SRMP.

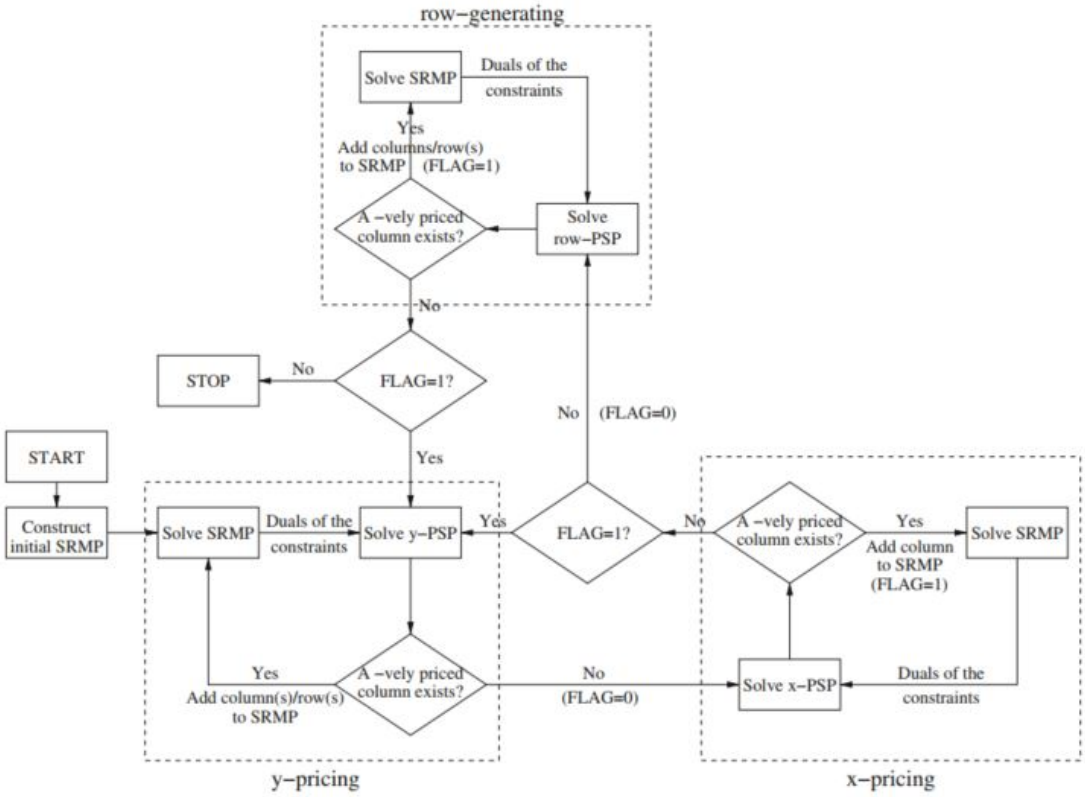


Figura 1.2: Esquema propuesto en Muter et al. (2013) para problemas CDR

En el trabajo de Feillet et al. (2010) se propone otra metodología para tratar con problemas CDR. Esta es más amplia que la propuesta por Muter et al. (2013), sin embargo requiere de más trabajo por parte de quienes deseen aplicarla puesto que tiene que ser adaptada y rediseñada para cada aplicación específica, en contraste con la anteriormente descrita en donde sólo debe reformularse según la forma canónica expuesta y cumplir con los 3 supuestos requeridos para asegurar un correcto funcionamiento.

Lo propuesto por Feillet et al. (2010) es un método de tres pasos que en ningún caso son simples, en el cual se debe generar, a partir de una solución primal factible, de una solución al dual del problema maestro (no restringido). Los autores toman un problema genérico con un maestro denominado por $MP\left(\Omega\right)$, en donde Ω representa el set de columnas factibles

del problema maestro (que en el contexto de generación de columnas su cardinalidad es muy grande). El maestro restringido es representado por $MP(\Omega')$, en donde $\Omega' \subseteq \Omega$ y las soluciones y valores objetivo de dichos problemas son representados por X^*, $X^{*'}$, $z^*_{MP(\Omega)}$ y $z^*_{MP(\Omega')}$, respectivamente.

Para los problemas duales de $MP(\Omega)$ y $MP(\Omega')$, se designan $D(\Omega)$ y $D(\Omega')$. Las soluciones y valores de dichos duales son representados por Π^*, $\Pi^{*'}$, $z^*_{D(\Omega)}$ y $z^*_{D(\Omega')}$. Del teorema de dualidad fuerte sabemos que, en el óptimo, $z^*_{MP(\Omega')} = z^*_{D(\Omega')}$ y $z^*_{MP(\Omega)} = z^*_{D(\Omega)}$.

El criterio de optimalidad utilizado para asegurar que $z^*_{MP(\Omega)} = z^*_{MP(\Omega')}$ y para detener el algoritmo de generación de columnas es tal que la solución dual del maestro restringido ($\Pi^{*'}$) es factible para el dual del maestro completo ($D(\Omega)$). Lo anterior se puede ver fácilmente puesto que la solución a $D(\Omega')$ define también una solución (no necesariamente factible) del problema $D(\Omega)$ puesto que ambos problemas están compuestos por las mismas variables. Luego, y tomando en cuenta de que el dual es un problema de maximización para un primal de minimización, tendremos que $z^*_{D(\Omega)} \leq z^*_{D(\Omega')}$, implicando por dualidad fuerte que en el óptimo $z^*_{D(\Omega')} = z^*_{D(\Omega)}$ y, en consecuencia, $z^*_{MP(\Omega')} = z^*_{MP(\Omega)}$.

En el caso de problemas CDR el criterio anterior se difumina debido a que como estos problemas crecen vertical y horizontalmente en cada iteración del algoritmo de generación de columnas, los problemas $D(\Omega')$ y $D(\Omega)$ ya no están definidos sobre el mismo set de variables duales, esto por al menos una variable dual. En consecuencia no se puede decir directamente que $D(\Omega')$ es una relajación de $D(\Omega)$ y la condición de optimalidad discutida en el párrafo anterior pierde sentido. La propuesta de Feillet et al. (2010) para remediar lo anterior se puede resumir en los siguientes pasos:

i. Desde una solución $X^{*'}$ del $MP(\Omega')$ se construye una solución X al problema maestro $MP(\Omega)$ tal que $z_{MP} = z^{*'}_{MP(\Omega')}$.

ii. Desde la solución $\Pi^{*'}$ del dual $D(\Omega')$ se construye una solución Π^*, no necesariamente factible, para $D(\Omega)$, de modo que $z_D = z^*_{D(\Omega')}$, y $z_D = z_{MP}$.

iii. Si la solución creada, Π, el criterio de optimalidad es alcanzado y se detiene el algoritmo. De otra forma, al menos un elemento del set $\Omega \setminus \Omega'$ tiene su restricción en el problema $D(\Omega)$ y es candidata a entrar en el set Ω'.

Sin embargo, y como fue expresado anteriormente, no hay una forma única de generar las soluciones X y Π, y debe especificarse una en cada aplicación en donde se desee emplear la metodología.

Para el correcto funcionamiento del algoritmo es necesario que se cumpla una condición implícita en el punto tres y es que el método debe poder encontrar una columna en el set $\Omega \setminus \Omega'$ dentro de la infactibilidad de la solución creada Π (cuando es infactible). De no poder hacerlo ninguna nueva columna ingresará al set Ω' y el método no podrá converger al óptimo.

1.3. Solving Constraint Integer Programs (SCIP)

SCIP (Achterberg et al., 2008) es un software gratuito para uso académico y de aplicaciones no comerciales, creado para resolver *Constraint Integer Programs* (CIPs). Desarrollado en el Konrad-Zuse-Zentrum für Informationestechnik Berlin (ZIB) desde 2001, y escrito en **C**, es el sucesor del solver para MIPs SIP (Martin, 1999), adoptando algunas de sus ideas y algoritmos. En Achterberg (2007) y Achterberg (2009) se pueden encontrar descripciones detalladas de cómo funciona este *framework* de resolución de CIPs.

Este representa un nuevo paradigma de resolución de CIPs, que integra técnicas de modelamiento y resolución de *constraint programming* (CP), *mixed integer programming* (MIP) y *satisfiability* (SAT). SCIP provee la infraestructura para implementar algoritmos flexibles de búsqueda tipo *B& B* e incluye un vasto set de algoritmos que permiten controlar la búsqueda a través del árbol de ramificación. Sin embargo, el corazón de SCIP esta en la flexibilidad que entrega para implementar reglas definidas por el usuario, a través de *plugins*, de forma sencilla vía de objetos de llamada (*callback objects*) y sin comprometer la estructura de resolución de CIPs. Dichos *plugins* pueden ser incorporadas al *framework* de SCIP y ser llamados durante el proceso de resolución. Esta arquitectura y flexibilidad la hacen una herramienta ideal para la programación y solución de problemas *Branch & Price* y *Branch & Cut & Price*.

Descritos en detalle en Achterberg (2007), se describen los principales *plugins* utilizados por SCIP. Los objetos principales de SCIP son los *constraint handlers*. Cada uno de estos representa un tipo de restricción, ej. restricciones lineales o *knspsack* (mochila), y provee métodos para controlar dichas restricciones y almacenar su información. La tarea principal de estas plugins es chequear la factibilidad de las soluciones con respecto a la clase de restricciones a las que pertenece y forzar de que dichas soluciones cumplan con todas y cada una de las restricciones de dicha clase. Estos *plugins*, además, pueden proveer de métodos adicionales específicos a la clase de restricciones que manejan para las fases de solución del problema.

Los siguientes *plugins* son aquellos relacionados al esquema de *B&B*, como reglas de ramificación y selección de nodos, cuyos fines ya fueron explicados anteriormente. También se pueden especificar como *plugin* reglas de *domain propagation*, también llamadas de preprocesamiento de nodos, que son ocupadas en cada nodo del árbol de ramificación. Los *relaxation handlers* son *plugins* que permiten deseñar e implementar reglas de cálculo de cotas primales y duales, que pueden ser usadas en conjunto, o en vez, de las relajaciones lineales. Por otro lado están las heurísticas primales, que tratan de encontrar soluciones factibles a lo largo del árbol, y separadores de planos cortantes, que buscan desigualdades válidas que buscan cortar una relajación lineal o una solución arbitraria.

Finalmente están los *variable pricers*, corazón de todo algoritmo de generación de columnas, que detallan la forma de búsqueda e inclusión de variables al problema durante su resolución.

La última versión de SCIP (Gamrath et al., 2020) incorpora ya varias opciones para cada uno de estos *plugins* para la resolución de MIPs, con la excepción de los *variable pricers* que en toda aplicación deben ser específicamente diseñados.

Capítulo 2

Definición del Problema

El problema estudiado en esta tesis es una variante particular de los clásicos problemas de *Inventory and Routing*. Como en todo IRP es necesario definir para cada cada cliente (cajero automático) cuándo se debe visitar, cuánto producto (efectivo) se debe entregar y la ruta que cada vehículo de reparto debe realizar para entregar los productos a sus clientes, asegurando suficiente disponibilidad de este para satisfacer la demanda de cada cliente en el intervalo de tiempo planificado. Esto minimizando los costos de entrega, inventario y otros, sujeto a diversas restricciones.

Adicional a las naturales dificultades y restricciones de un IRP, el problema de esta tesis tiene características particulares, no tan sólo en relación a otros IRP sino también a otros problemas de ruteo e inventario de cajeros automáticos. Estas son:

(i) Se permiten *stock-outs* en los ATM, es decir, todo cajero automático puede quedar sin efectivo en cierto período, en este caso la demanda no satisfecha se pierde. El costo de caer en tal estado se manifiesta de dos formas: fijo y variable. El primero es gatillado por tan sólo quedar en *stock-out*, mientras que el segundo es proporcional a la demanda no satisfecha en el periodo y cajero afectado, como se detalla en la Figura 2.1. Los trabajos de Van Anholt et al. (2015) y Koc et al. (2018) no contemplan este tipo de penalidades mientras Larrain et al. (2017) sólo contempla costos fijos por quiebre.

(ii) Si bien se permiten los *stock-out* en los cajeros, se presentan restricciones que aseguran un nivel mínimo de servicio. En particular estas restricciones son tres: ningún cajero puede estar más del 50 % del horizonte de tiempo en quiebre, en ningún periodo pueden estar más del 50 % de los cajeros en *stock-out* y del total de combinaciones cajero-periodo no más del 25 % puede presentar *stock-outs*. Esto viene a impedir que una solución factible al problema sea dejar todos los cajeros, en todos sus periodos sin efectivo.

    Si por el contrario, no se permitiera el quiebre de inventario, tal como pasa en Van Anholt et al. (2015) y Koc et al. (2018), esto implicaria que el nivel de servicio requerido es de un 100 %. Por otro lado vemos que en Larrain et al. (2017) se permiten los quiebres de inventario, sin embargo no existe requerimiento alguno respecto al nivel de servicio,

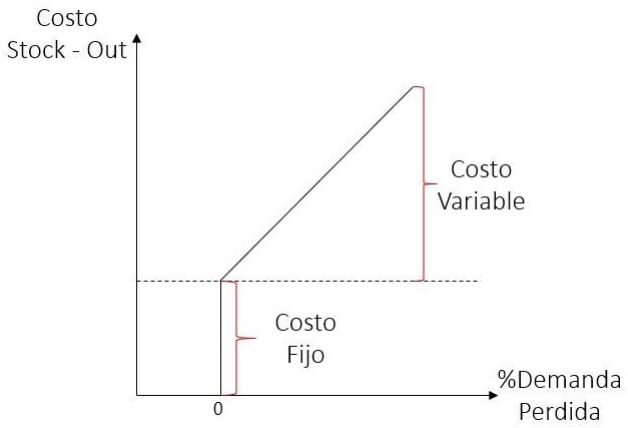


Figura 2.1: Costos de quiebre en un Cajero - Periodo

permitiendo de esta forma que se puedan dejar todos los cajeros sin circulante.

(iii) Las entregas de efectivo a cada cajero se realizan en *cassette*, estos son contenedores sellados de dinero y, en consecuencia, al hacer las entregas en cada cajero se debe reemplazar el *cassette* antiguo por el nuevo, independiente de si el antiguo todavía tenía efectivo en él o no. Así el monto entregado por un *cassette* no se suma al monto de efectivo existente en el cajero al inicio del período. En Larrain et al. (2017) se sigue la misma idea, mientras que tanto en Van Anholt et al. (2015) como en Koc et al. (2018) las entregas son adicionadas al inventario existente.

Como supuesto se asume que el monto de efectivo contenido en el *cassette* de menor tamaño es suficiente para cubrir la demanda de un periodo en cualquier cajero. Por otro lado, las visitas a cada cajero se encuentran restringidas a un máximo de una visita por período. En consecuencia, no es posible que un cajero que ha sido visitado pueda caer en *stock-out* en el tiempo remanente del periodo.

(iv) Hay un número finito de tipos de *cassette*, estos representan montos de efectivo diferentes. Adicionalmente, y por motivos de seguridad, cada camión tiene un monto máximo que puede entregar por ruta, esto limita la cantidad de cajeros que es posible visitar en cada ruta.

(v) Se incorpora el momento exacto de visita a cada ATM en las rutas realizadas. Por lo que no es trivial el orden en que se visitan los cajeros que componen una ruta determinada. Entre más tardía la visita dentro de una ruta más inventario tendrá que tener, al inicio del período, el ATM para cubrir la demanda hasta el momento de visita. En la mayoría de los IRP se asume que el momento de reposición de inventario ocurre al inicio de cada periodo, lo que implica que en la práctica se subestiman la cantidad de quiebres en que se incurre.

(vi) Complementando el punto anterior, la demanda no se consume por completo en algún momento de cada período (inicio o fin), sino que linealmente durante la duración de cada período. Por ejemplo si para un cajero cualquiera, en un determinado período su demanda es 50 unidades, a la mitad de ese período se habrán consumido exactamente

25 unidades.

(vii) Se facilita la salida retrasada de camiones, es decir, no es necesario que los camiones a los que se le asigna una ruta deban salir al inicio del período. Sin embargo, el tiempo de retraso en la salida más el tiempo que el camión demora en completar la ruta debe ser inferior al largo del período.

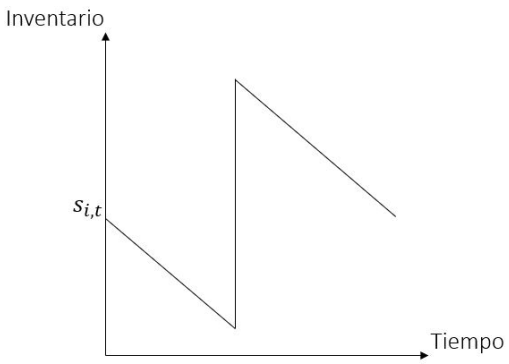
Como en otros IRP, existen restricciones de capacidad. Las dos primeras se deben a la seguridad y los seguros asociados a la operación y transporte de efectivo, la primera impone un límite en la duración de las rutas (tiempo máximo que un camión puede estar fuera del depot repartiendo efectivo) y la cantidad total que un camión puede transportar por ruta. La tercera restricción de capacidad hace referencia a la cantidad total de circulante que un cajero puede almacenar, sin embargo, en este trabajo tal límite es igual al tamaño del *cassette* más grande.

Los puntos (v) a (vii) son elementos no encontrados en la literatura revisada sobre IRPs relacionados al manejo de cajeros automáticos. Por lo demás, al unirlos con los puntos (i) y (ii) implican una dinámica particular en los inventarios de los cajeros. Sea s_{it} el inventario en el cajero i al inicio del periodo t, o_{it} el monto de demanda no satisfecha y q_{it} el monto entregado en tal cajero-periodo, la dinámica de inventario y quiebre se puede modelar de la siguiente forma:

$$s_{i,t+1} = \begin{cases} \max\{s_{it} - \lambda_{it}, 0\} & \text{Si no hay visita} \\ q_{it} - \lambda_{it}^{+} & \text{si hay visita} \end{cases} \tag{2.1}$$

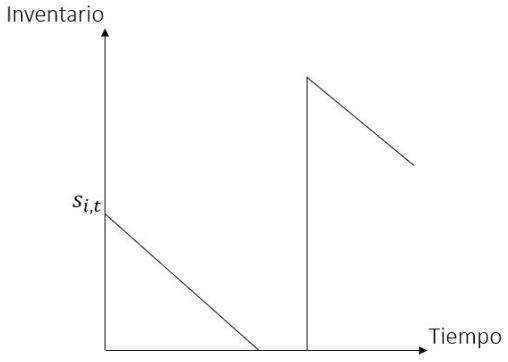
$$o_{it} = \begin{cases} \max\{\lambda_{it} - s_{it}, 0\} & \text{si no hay visita} \\ \max\{(\lambda_{it} - \lambda_{it}^{+}) - s_{it}, 0\} & \text{si hay visita} \end{cases} \tag{2.2}$$

donde λ_{it} es la demanda del cajero i en el periodo t y λ_{it}^{+} es la demanda realizada después de que el cajero i ha sido visitado. En cada variable podemos ver al menos cuatro estados que dependen de la realización de una visita al cajero y si este cae en *stock-out* durante el período. Primero revisamos el caso en que se realiza una vista al cajero en cuestión:



$$\begin{aligned} s_{i,t+1} &= q_{i,t} - \lambda_{it}^{+} \\ o_{it} &= 0 \end{aligned}$$

Figura 2.2: Evolución del inventario con visita y sin quiebre

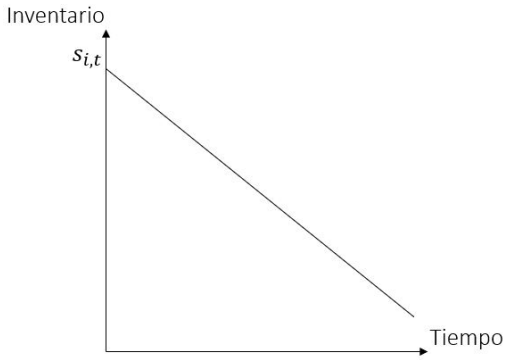


$$\begin{aligned} s_{i,t+1} &= q_{it} - \lambda_{it}^{+} \\ o_{it} &= \left(\lambda_{it} - \lambda_{it}^{+}\right) - s_{it} \end{aligned}$$

Figura 2.3: Evolución del inventario con visita y con quiebre

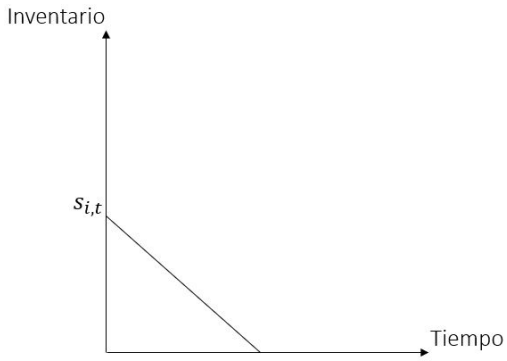
Las Figuras 2.2 y 2.3 muestran la evolución del inventario de un cajero en un periodo determinado en el cual este es visitado. En la primera de las imágenes el cajero es visitado antes de quedarse sin efectivo y el valor del inventario al final del periodo es igual al monto entregado menos la cantidad de demanda realizada posterior a la entrega. Tal como la Figura 2.3 muestra el inventario cae linealmente en el tiempo puesto que la demanda no es realizada de forma instantánea sino a tasa constante a lo largo del periodo. La Figura 2.3 en cambio muestra un quiebre de stock antes de que la visita se realice, el valor de inventario al final del periodo es el mismo que en el figura anterior, pero el nivel de quiebre ya no es cero y se ha generado tanto un costo fijo por la falta de circulante como uno variable, acorde al nivel de demanda perdida. Esto viene reforzar la idea de que en este problema el orden de visita de las rutas no es irrelevante, no solo en términos de costo de transporte sino también en términos de inventario.

En las figuras 2.4 y 2.5 vemos la evolución del inventario cuando este no es visitado durante el periodo. En la primera no hay quiebre y el inventario final es simplemente el inventario inicial menos la demanda del periodo. Mientras, en la segunda figura, se genera un quiebre, el inventario final es cero y el nivel de quiebre es igual a la demanda del periodo menos el inventario inicial.



$$\begin{aligned} s_{i,t+1} &= s_{it} - \lambda_{it}^{+} \\ o_{it} &= 0 \end{aligned}$$

Figura 2.4: Evolución del inventario sin visita y sin quiebre



$$\begin{aligned} s_{i,t+1} &= 0 \\ o_{it} &= \lambda_{it} - s_{it} \end{aligned}$$

Figura 2.5: Evolución del inventario sin visita y con quiebre

Como últimos detalles del problema, la red de cajeros automáticos es representada mediante un grafo dirigido de costos simétricos, donde la distancia entre dos cajeros esta calculada como la distancia euclidiana entre ambos. Los tiempos de viaje entre dos cajeros, t_{ij}, se componen de dos partes: El tiempo de viaje entre i y j, y el tiempo de que toma reemplazar el *cassette* del cajero j, tiempo que se asume igual para todo cajero.

La demanda en cada cajero, para cada uno de los periodos en el horizonte de tiempo a planificar se asume conocida y determinística. También se conocen los inventarios de cada cajero al inicio del horizonte de planificación y se asume que el depot (de donde salen y donde deben llegar todos los camiones) tiene suficiente inventario para cubrir todos los envíos planificados.

Capítulo 3

Modelamiento y Estrategia de Resolución

En este capítulo se desarrolla la modelación del problema, el esquema de resolución y su implementación. Tal como exponen Coelho et al. (2014) y Coelho and Laporte (2014) los problemas de ruteo e inventario son de dificultad NP-hard debido a que ellos incluyen en sí problemas de ruteo de vehículos y, en consecuencia, muchas de las soluciones propuestas en la literatura son heurísticas. Sin embargo acá se plantea un algoritmo de solución exacta, específicamente, un esquema de *Branch & Price* cuya descomposición en maestro y sub-problemas permite un mejor manejo de la gran cantidad de rutas posibles para el problema. Estas rutas serán entregadas por los sub-problemas a medida que el maestro las necesite, evitando la necesidad de lidiar con todas las rutas factibles posibles simultáneamente.

En el capítulo anterior vimos como el problema de esta tesis difiere del resto de la literatura asociada al manejo y operación de ATMs vía IRPs. La metodología empleada en este trabajo también difiere de las utilizadas anteriormente. Van Anholt et al. (2015) utiliza un algoritmo de *Branch & Cut* complementado con una heurística de clusterización que es utilizada previa al algoritmo para poder atacar instancias más grandes, Larrain et al. (2017) utiliza una heurística de *Variable MIP Neighborhood Search* (VMNS) y Koc et al. (2018) combina un método de sistema de información geográfica (GIS-based solution) en conjunto de una heurística de búsqueda tabú para resolver el problema de un banco en específico en la ciudad de Gaziantep, Turquía.

La primera parte del capítulo se dedica a la modelación del maestro, su dual, el sub-problema de generación de columnas y sus dificultades. La segunda parte habla del esquema de *Branch & Price*, la estrategia de ramificación y selección de nodos empleada. Finalmente la tercera parte detalla la implementación del esquema y su programación.

3.1. Problema Maestro

El problema es de un operador que tiene bajo su administración una red de cajeros automáticos la cual debe abastecer de efectivo según sea necesario con el fin de minimizar los costos totales de operación. Esta red es representada por un set de nodos I en donde el nodo

0 representa el depot y el resto de ellos, $I' = I \setminus \{0\}$, representa cada uno de los cajeros. Se considera un horizonte de tiempo ρ, de largo $\overline{T}$, donde cada periodo $t \in T$ el operador debe decidir qué cajeros visitar, qué *cassette* utilizar en la reposición, en qué rutas y el orden en que cada uno será visitado.

Se define el set R_{it} como el set de rutas (factibles) que visitan al cajero $i \in I'$ en el periodo $t \in T$, en consecuencia $R = \cup_{it} R_{it}$ es el set de todas las rutas presentes en el maestro. A continuación se detallan todos los parámetros y variables del problema maestro y sus definiciones.

3.1.1. Parámetros del Problema Maestro

M_i: Valor suficientemente grande. En el caso de este problema se le asigna un valor igual a la cantidad máxima que puede ser entregada al cajero i.

Q_{irt}: Cantidad de dinero que la ruta r deja en el cajero i en el período t.

λ_{it}: Demanda del cajero i en el período t.

$\overline{\Lambda}_{irt}$: Demanda por efectivo en el cajero i luego de que la ruta r ha hecho su entrega en el periodo t, dado que la ruta se ejecuta lo más temprano posible.

c_{rt}: Costo de la ruta r en el período t.

d_{rt}: Duración de la ruta r en el periodo t, como fracción del período.

g_i: Costo unitario por demanda perdida en el cajero i.

h_{it}: Costo de oportunidad unitario del inventario en el cajero i en el período t (*holding cost*).

k_i: Costo fijo por quiebre en el cajero i.

θ_i: Inventario inicial en el cajero i.

Es importante establecer que los parámetros Q_{irt}, $\overline{\Lambda}_{irt}$, c_{rt} y d_{rt} son parámetros obtenidos del sub-problema de generación de columnas. En el caso de $\overline{\Lambda}_{irt}$ el valor toma en cuenta que la salida del camión que toma aquella ruta lo hace al inicio del período puesto que en el maestro es posible retrasar la salida de los camiones en tal ruta, modificando así la demanda remanente posterior a la visita del camión.

3.1.2. Variables del Problema Maestro

x_{rt}: Variable binaria que indica si se utiliza la ruta r en el período t, o no.

y_{it}: Variable binaria que indica si el cajero i incurre en quiebre de stock en el período t, o no.

f_{rt}: Variable continua que indica el momento en que la ruta r inicia su recorrido en el período t, como fracción del período.

s_{it}: Variable continua, no negativa, que indica el nivel de inventario del cajero i al inicio del período t.

o_{it}: Variable continua, no negativa, que indica el nivel de quiebre de stock del cajero i en el período t.

3.1.3. Formulación Problema Maestro

Antes de entrar en el modelo, recordemos que la variable s_{it} toma en cuenta el nivel de inventario al **inicio** del periodo, por tanto agregamos para dichas variables un periodo más de forma de incluir el estado del inventario al final del horizonte de planificación y tratar de evitar stocks outs en las condiciones iniciales de la siguiente planificación. Entonces, se define $T' = T \cup \{\rho + 1\}$ como el set de periodos disponibles para las variables s_{it},o_{it} y y_{it}.

La función objetivo del problema se define como:

$$\text{mín} \quad \sum_{r\in R}\sum_{t\in T} c_{rt}x_{rt} + \sum_{i\in I'}\sum_{t\in T'} h_{it}s_{it} + \sum_{i\in I'}\sum_{t\in T'} g_i o_{it} + \sum_{i\in I'}\sum_{t\in T'} k_i y_{it} \tag{3.1}$$

Sujeto a

$$\begin{aligned}
-s_{it} + s_{i,t+1} - M_i \sum_{r\in R_{it}} x_{rt} - \lambda_{it} y_{it} &\leq -\lambda_{it} \qquad \forall i \in I', t \in T && (3.2)\\
s_{it} - s_{i,t+1} - M_i \sum_{r\in R_{it}} x_{rt} &\leq \lambda_{it} \qquad \forall i \in I', t \in T && (3.3)\\
-s_{i,t+1} + \sum_{r\in R_{it}} \left(Q_{irt} - \overline{\Lambda}_{irt}\right) x_{rt} + \lambda_{it} \sum_{r\in R_{it}} f_{rt} &\leq 0 \qquad \forall i \in I', t \in T && (3.4)\\
s_{i,t+1} + \sum_{r\in R_{it}} \left(-Q_{irt} + \overline{\Lambda}_{irt} + M_i\right) x_{rt} - \lambda_{it} \sum_{r\in R_{it}} f_{rt} &\leq M_i \qquad \forall i \in I', t \in T && (3.5)\\
s_{i,t+1} - \sum_{r\in R_{it}} \left(Q_{irt} - \overline{\Lambda}_{irt}\right) x_{rt} - \lambda_{it} \sum_{r\in R_{it}} f_{rt} + M_i y_{it} &\leq M_i \qquad \forall i \in I', t \in T && (3.6)\\
-s_{it} - \sum_{r\in R_{it}} \overline{\Lambda}_{irt} x_{rt} + \lambda_{it} \sum_{r\in R_{it}} f_{rt} - o_{it} &\leq -\lambda_{it} \qquad \forall i \in I', t \in T' && (3.7)\\
-\lambda_{it} y_{it} + o_{it} &\leq 0 \qquad \forall i \in I', t \in T && (3.8)\\
\sum_{r\in R_{it}} x_{rt} &\leq 1 \qquad \forall i \in I', t \in T && (3.9)\\
-\left(1 - d_{rt}\right) x_{rt} + f_{rt} &\leq 0 \qquad \forall r \in R, t \in T && (3.10)\\
\sum_{i\in I'} y_{it} &\leq 0{,}5 \cdot |I'| \qquad \forall t \in T' && (3.11)\\
\sum_{t\in T} y_{it} &\leq 0{,}5 \cdot |T'| \qquad \forall i \in I' && (3.12)\\
\sum_{i\in I'}\sum_{t\in T} y_{it} &\leq 0{,}25 \cdot |T'| \cdot |I'| && (3.13)\\
s_{i,1} &= \theta_i \qquad \forall i \in I' && (3.14)\\
x_{rt} &\in \{0,1\} \qquad \forall r \in R, t \in T && (3.15)\\
y_{it} &\in \{0,1\} \qquad \forall i \in I', t \in T' && (3.16)\\
f_{rt} &\geq 0 \qquad \forall r \in R, t \in T' && (3.17)\\
s_{it} &\geq 0 \qquad \forall i \in I', t \in T' && (3.18)\\
o_{it} &\geq 0 \qquad \forall i \in I', t \in T' && (3.19)
\end{aligned}$$

En la ecuación (3.1) tenemos la función objetivo que representa los costos de operación del administrador, contiene los costos incurridos por las rutas empleadas, el costo de oportunidad del circulante en los inventarios de los cajeros y los costos de *stock-out*.

Las restricciones (3.2) a (3.8) nos ayudan a modelar la dinámica de inventario y nivel de quiebre descritas en las ecuaciones (2.1) y (2.2) del capítulo anterior. En lo siguiente veremos cómo es que esos casos se realizan en cada una de los 4 escenarios posibles para un cajero cualquiera en un periodo específico:

1. **No hay visita ni quiebre:**

$s_{i,t+1}$: Las restricciones (3.2) y (3.3) fijan la variable de inventario en:

$$s_{i,t+1} \leq s_{it} - \lambda_{it} \tag{3.20}$$
$$s_{i,t+1} \geq s_{it} - \lambda_{it} \tag{3.21}$$

o_{it}: La restricción (3.8) restringe la variable en:

$$o_{it} \leq 0 \tag{3.22}$$

2. **No hay visita pero sí quiebre:**

$s_{i,t+1}$: Las restricciones (3.4) y (3.6) fijan la variable en:

$$s_{i,t+1} \geq 0 \tag{3.23}$$
$$s_{i,t+1} \leq 0 \tag{3.24}$$

o_{it}: La restricción (3.7) pone un máximo a la variable en:

$$o_{it} \geq \lambda_{it} - s_{it} \tag{3.25}$$

Como es un problema de minimización de costos el valor de o_{it} se quedará en la cota mínima de la expresión (3.25).

3. **Hay visita, pero no quiebre:**

$s_{i,t+1}$: Las restricciones (3.4) y (3.5) fijan la variable en:

$$s_{i,t+1} \geq \sum_{r \in R_{it}} \left(Q_{irt} - \overline{\Lambda}_{irt}\right) x_{rt} + \lambda_{it} \sum_{r \in R_{it}} f_{rt} \tag{3.26}$$
$$s_{i,t+1} \leq \sum_{r \in R_{it}} \left(Q_{irt} - \overline{\Lambda}_{irt}\right) x_{rt} + \lambda_{it} \sum_{r \in R_{it}} f_{rt} \tag{3.27}$$

Donde la expresión a la derecha de cada ecuación es la cantidad entregada por el camión en la visita (*cassette*) menos la demanda servida posterior a dicha visita, el producto de la demanda (λ_{it}) con la variable f_{rt} viene a ajustar dicha demanda remanente cuando el vehículo comienza su ruta con retraso.

o_{it}: La restricción (3.8) pone un máximo a la variable en:

$$o_{it} \leq 0 \tag{3.28}$$

4. **Hay visita y quiebre:** En este caso el quiebre debe darse antes de la visita puesto que uno de los supuestos es que el *cassette* de menor valor cubre la demanda de cualquier cajero para cualquier periodo, entonces el cajero no puede quedarse sin inventario en tal periodo una vez que ha sido visitado.

$s_{i,t+1}$: Las restricciones (3.4) y (3.5) fijan la variable en:

$$s_{i,t+1} \geq \sum_{r \in R_{it}} \left(Q_{irt} - \overline{\Lambda}_{irt}\right) x_{rt} + \lambda_{it} \sum_{r \in R_{it}} f_{rt} \tag{3.29}$$

$$s_{i,t+1} \leq \sum_{r \in R_{it}} \left(Q_{irt} - \overline{\Lambda}_{irt}\right) x_{rt} + \lambda_{it} \sum_{r \in R_{it}} f_{rt} \tag{3.30}$$

o_{it}: La restricción (3.7) pone un mínimo a la variable en:

$$o_{it} \geq \left(\lambda_{it} - \sum_{r \in R_{it}} \overline{\Lambda}_{irt} x_{rt} + \lambda_{it} \sum_{r \in R_{it}} f_{rt}\right) - s_{it} \tag{3.31}$$

La expresión en paréntesis es equivalente a la demanda realizada antes de que se realizara la vista al cajero.

La restricción (3.9) indica que cada cajero sólo puede ser visitado, como máximo, una vez por periodo. La restricción (3.10) restringe el máximo retraso con que puede iniciar cada ruta, tomando en cuenta el tiempo que demora cada ruta en completarse (d_{rt}). Las restricciones (3.11) a (3.13) establecen el mínimo nivel de servicio que la operación debe tener en términos de quiebres por cajero, periodo y horizonte total de planificación. La restricción (3.14) indica el inventario inicial en cada cajero. Finalmente las expresiones (3.15) a (3.19) detallan la naturaleza de las variables del modelo.

3.2. Problema Dual

La Tabla 3.1 muestra las variables duales del problema maestro y su relación con las restricciones de este, con aquella información es posible extraer el problema dual. En este encontraremos la expresión de costos reducidos para la variable de interés, x_{rt}.

Tabla 3.1: Restricciones maestro y variables duales

Restricción	Dual	Restricción	Dual	Restricción	Dual
(3.2)	α_{it}	(3.3)	β_{it}	(3.4)	γ_{it}
(3.5)	ϕ_{it}	(3.6)	ζ_{it}	(3.7)	δ_{it}
(3.8)	ν_{it}	(3.9)	μ_{it}	(3.10)	η_{rt}
(3.11)	φ_t^A	(3.12)	φ_i^B	(3.13)	φ^C
(3.14)	π_i				

Adicional a la información contenida en la Tabla 3.1, se define al set κ_{rt} como el set de todos los cajeros contenidos en la ruta r del periodo t. Con todo lo anterior presenta el problema dual del problema maestro, el cuál queda de la siguiente forma:

$$\text{máx} \quad \sum_{i\in I',t\in T}\left[M_i\left(\phi_{it}+\zeta_{it}\right)+\lambda_{it}\left(\beta_{it}-\alpha_{it}-\delta_{it}\right)+\mu_{it}+0{,}5|T|\varphi_i^B\right] + \sum_{i\in I'}\theta_i\pi_i+0{,}5|I'|\sum_{t\in T}\varphi_t^A+0{,}25|I'||T|\varphi^C \tag{3.32}$$

Sujeto a

$$\sum_{i\in\kappa_{rt}}\left[Q_{irt}\left(\gamma_{it}-\phi_{it}-\zeta_{it}\right)-\overline{\Lambda}_{irt}\left(\gamma_{it}-\phi_{it}-\zeta_{it}+\delta_{it}\right)-M_i\left(\alpha_{it}+\beta_{it}-\phi_{it}\right)+\mu_{it}\right] - \left(1-d_{rt}\right)\eta_{rt}\le c_{rt} \qquad \forall r\in R, t\in T \tag{3.33}$$

$$\beta_{i,1}-\alpha_{i,1}-\delta_{i,1}+\pi_{i,1}\le h_{i,1} \qquad \forall i\in I', t=1 \tag{3.34}$$

$$\sum_{i\in I'\setminus\{1\}}\left(\beta_{it}-\beta_{i,t-1}\right)-\left(\alpha_{it}-\alpha_{i,t-1}\right)-\delta_{it}+\phi_{i,t-1}+\zeta_{i,t-1}-\gamma_{i,t-1} \le h_{it} \qquad \forall i\in I', t\in T'\setminus\{1\} \tag{3.35}$$

$$\sum_{i\in\kappa_{rt}}\lambda_{it}\left(\gamma_{it}+\delta_{it}-\phi_{it}-\zeta_{it}\right)+\eta_{rt} \le 0 \quad \forall r\in R, t\in T \tag{3.36}$$

$$M_i\zeta_{it}-\lambda_{it}\left(\alpha_{it}+\nu_{it}\right)+\varphi^A+\varphi^B+\varphi^C \le k_i \quad \forall i\in I', t\in T' \tag{3.37}$$

$$\nu_{it}-\delta_{it} \le g_i \quad \forall i\in I', t\in T' \tag{3.38}$$

$$\alpha_{it} \le 0 \quad \forall i\in I', t\in T \tag{3.39}$$

$$\beta_{it} \le 0 \quad \forall i\in I', t\in T \tag{3.40}$$

$$\gamma_{it} \le 0 \quad \forall i\in I', t\in T \tag{3.41}$$

$$\phi_{it} \le 0 \quad \forall i\in I', t\in T \tag{3.42}$$

$$\zeta_{it} \le 0 \quad \forall i\in I', t\in T \tag{3.43}$$

$$\delta_{it} \le 0 \quad \forall i\in I', t\in T \tag{3.44}$$

$$\nu_{it} \le 0 \quad \forall i\in I', t\in T \tag{3.45}$$

$$\mu_{it} \le 0 \quad \forall i\in I', t\in T \tag{3.46}$$

$$\pi_i \le 0 \quad \forall i\in I' \tag{3.47}$$

$$\eta_{rt} \le 0 \quad \forall r\in R, t\in T \tag{3.48}$$

La Tabla 3.2 muestra la relación entre restricciones duales y las variables del maestro.

Tabla 3.2: Relación entre restricciones duales y variables del maestro

Restricción Dual	(3.33)	(3.36)	(3.37)	(3.38)	(3.34) y (3.35)
Variable Primal	$x_{r,t}$	$f_{r,t}$	$y_{i,t}$	$o_{i,t}$	$s_{i,t}$

La restricción (3.33) nos ayudará a obtener la función objetivo del sub-problema de generación de columnas, puesto que representa la expresión de costos reducidos de las variables x_{rt}.

3.3. Sub-problema de generación de columnas

Se procede a formular el problema de generación de columnas, este está encargado de generar rutas factibles de costo reducido negativo al problema maestro restringido. Antes de comenzar definimos los parámetros y variables a utilizar:

3.3.1. Parámetros del sub-problema

L_p: Cantidad de dinero que carga la gaveta tipo p. El set P es el set de todos los tipos de *cassette*.

$\overline{T}$: Duración total del período.

t_{ij}: Tiempo que demora ir desde el cajero i al cajero j, más el tiempo que toma atender el cajero i.

c_{ij}: Costo de ir desde el cajero i al cajero j.

$\overline{Q}$: Cantidad máxima de dinero que puede llevar en una ruta.

$\bar{t}$: Duración máxima que puede tener una ruta.

El grafo asociado a la red de cajeros automáticos posee el set, ya conocido, I de nodos (cajeros y depot) y un set $A = \{ij \in I \times I : i \neq j\}$ de arcos.

3.3.2. Variables sub-problema

y_{ij}: Variable binaria que indica si se viaja desde el cajero i al cajero j.

x_{ip}: Variable binaria que indica si al visitar el cajero i se entrega el *cassette* p.

w_i: Variable continua que indica el momento en que el camión deja el cajero i. Para w_0 indica el momento de llegada de vuelta al depot.

Del problema dual tomamos la restricción (3.33), que representa la función de costos reducidos de la variable x_{rt}. Esta es tomada como la función objetivo del sub-problema de generación de columnas.

$$\begin{aligned} &\text{mín} \quad c_{rt} + (1 - d_{rt})\,\eta_{rt} \\ &- \sum_{i \in \kappa_{rt}} \left[Q_{irt}\left(\gamma_{it} - \phi_{it} - \zeta_{it}\right) - \overline{\Lambda}_{irt}\left(\gamma_{it} - \phi_{it} - \zeta_{it} + \delta_{it}\right) - M_i\left(\alpha_{it} + \beta_{it} - \phi_{it}\right) + \mu_{it} \right] \end{aligned} \tag{3.49}$$

Los valores necesarios para introducir una nueva ruta al problema maestro, como podemos recordar, son Q_{irt}, $\overline{\Lambda}_{irt}$, c_{rt} y d_{rt}. A partir de las variables duales propuestas podemos obtener expresiones para estos parámetros:

$$c_{rt} = \sum_{i \in I, j \in I \setminus \{i\}} c_{ij} y_{ij} \tag{3.50}$$

$$\overline{\Lambda}_{irt} = \lambda_{it}\left(x_{ip} - \frac{w_i}{\overline{T}}\right) \tag{3.51}$$

$$Q_{irt} = L_p x_{ip} \tag{3.52}$$

$$(1 - d_{rt}) = \left(1 - \frac{w_0}{\overline{T}}\right) \tag{3.53}$$

Con lo anterior podemos reescribir la expresión (3.49) de forma de que se incorporen las variables propuestas para el sub-problema, la expresión queda de la siguiente manera:

$$\begin{aligned} \text{mín} \quad & \sum_{ij \in A} c_{ij} y_{ij} + \left(1 - \frac{w_0}{\overline{\overline{T}}}\right) \eta_{rt} \\ - \sum_{i \in I', p \in P} & \left[L_p \left(\gamma_{it} - \phi_{it} - \zeta_{it}\right) - M_i \left(\alpha_{it} + \beta_{it} - \phi_{it}\right) + \mu_{it}\right] x_{ip} \\ - \sum_{i \in I', p \in P} & \lambda_{i,t} \left(x_{i,p} - \frac{w_i}{\overline{\overline{T}}}\right) \left(\gamma_{it} - \phi_{it} - \zeta_{it} + \delta_{it}\right) \end{aligned} \tag{3.54}$$

Se puede apreciar que si bien hay valores en la función objetivo que poseen el sub-índice t no hay ninguna agrupación para este, la razón de aquello es debido a que tendremos un sub-problema de generación de columnas para cada período en el horizonte de tiempo.

Si bien tenemos una expresión en (3.54) para la función objetivo de los sub-problemas de generación de columnas, existe una dificultad adicional que se debe atender. Este hace referencia a la variable dual η_{rt}, puesto que si bien existen valores de esta variable para todas las rutas en el maestro reducido, estos valores son desconocidos para todas aquellas rutas ausentes en RMP (*restricted master problem*).

3.3.3. Dilema de column-dependent-rows

Los problemas de *column-dependent-rows* (CDR), según Muter et al. (2013), son aquellos en donde existe un set de restricciones que entrelazan variables, las cuáles o son muchas para ser incluidas directamente o sólo se pueden identificar cuando el set de variables completo ha sido descubierto, es decir sólo se pueden agregar a medida que las variables de interés (que están siendo agregadas paulatinamente) son introducidas al problema, tal como se explico anteriormente en el capítulo 1. Lo anterior implica que el maestro restringido crece horizontal y verticalmente durante la generación de columnas, o sea filas y columnas son generadas simultáneamente.

El problema anterior se da en el problema maestro, con la restricción (3.10), la cual limita el retraso con que cada ruta puede partir su recorrido, cuando es empleada. Al generar una nueva variable x_{rt} se conoce la necesidad de una restricción que limite el máximo retraso que esta puede tener en su salida, gracias al parámetro d_{rt}. Adicionalmente se hace imperativa la inclusión de una variable continua que tome en cuenta dicho retraso en el caso que se efectúe, f_{rt}. Es decir, con la creación de cada variable x_{rt} se debe agregar una restricción y una variable adicional al maestro restringido.

Lo recientemente expresado tiene consecuencias a la hora de ejecutar el sub-problema, ya que para aquellas variables x_{rt} ausentes en el RMP y, que por lo tanto no tienen una restricción tipo (3.10), no hay un valor para su variable dual η_{rt} que es parte de la expresión de costos reducidos con la que encontramos nuevas variables, (3.54). Esto conlleva el problema que, de aplicarse una generación de columnas genérica, los valores de costos reducidos encontrados

podrían estar sobrestimados, dejando fuera potenciales nuevas rutas factibles que mejorarían la solución óptima encontrada por el algoritmo.

Recordando, en Feillet et al. (2010) se propone un método que se puede resumir en los siguientes 3 puntos:

1. De la solución del maestro restringido se propone una solución al problema maestro completo tal que el valor de la función objetivo no cambie.

2. A partir de la solución del dual restringido se construye una solución, no necesariamente factible, al dual del problema maestro, tal que el valor de las soluciones no cambie.

3. De ser factible la solución dual construida, se cumple la condición de optimalidad y se detiene el algoritmo. De lo contrario existe una restricción dual violada y se agrega su variable primal asociada al maestro restringido, y se pasa a la siguiente iteración.

El primer punto de la lista es simple, se mantiene la solución obtenida en el maestro restringido y se asigna a toda variable no presente en este el valor cero. Para el segundo punto vemos que la variable conflictiva aparece en las restricciones (3.33) y (3.36), con la esta última restricción y la naturaleza de η_{rt} podemos obtener una expresión para dicha variable:

$$\eta_{rt} \leq \sum_{i \in \kappa_{rt}} \lambda_{it} \left(\phi_{it} + \zeta_{it} - \gamma_{it} - \delta_{it} \right) \quad \forall r \in R, t \in T \tag{3.55}$$

$$\eta_{r,t} \leq 0 \quad \forall r \in R, t \in T \tag{3.56}$$

Con lo anterior e introduciendo las variables del sub-problema propuestas, la expresión para η_{rt} queda:

$$\eta_{r,t} = \begin{cases} \sum\limits_{i \in I', p \in P} \lambda_{it} \left(\phi_{it} + \zeta_{it} - \gamma_{it} - \delta_{it} \right) x_{ip} & \text{Si } \ell \text{ se cumple} \\ 0 & \text{En cualquier otro caso} \end{cases} \tag{3.57}$$

Donde la condición ℓ se define como la desigualdad:

$$\sum_{i \in I', p \in P} \lambda_{i,t} \left(\phi_{i,t} + \zeta_{i,t} - \gamma_{i,t} - \delta_{i,t} \right) x_{ip} \leq 0 \tag{3.58}$$

Imponiendo esta condición creamos una solución al dual al maestro tal que el valor de la función objetivo del dual restringido no cambia y la condición de optimalidad es tal que se debe encontrar una solución dual restringida que sea factible para el dual no restringido, mediante la búsqueda de restricciones violadas del tipo (3.33). Con este cambio, la función objetivo del sub-problema queda de la siguiente forma:

$$\begin{aligned}\text{minimizar} \quad & \sum_{ij\in A} c_{ij}y_{ij} + \left(1 - \frac{w_0}{\overline{T}}\right) \sum_{i\in I', p\in P} \lambda_{i,t}\left(\phi_{i,t} + \zeta_{i,t} - \gamma_{i,t} - \delta_{i,t}\right) x_{ip} \\ & - \sum_{i\in I', p\in P} \left[L_p\left(\gamma_{it} - \phi_{it} - \zeta_{it}\right) - M_i\left(\alpha_{it} + \beta_{it} - \phi_{it}\right) + \mu_{it}\right] x_{ip} \\ & - \sum_{i\in I', p\in P} \lambda_{i,t}\left(x_{ip} - \frac{w_i}{\overline{T}}\right)\left(\gamma_{it} - \phi_{it} - \zeta_{it} + \delta_{it}\right)\end{aligned} \tag{3.59}$$

Al hacer este cambio, nos encontramos con un nuevo problema en la función objetivo del sub-problema. Este es que hay dos variables multiplicándose en ella: w_0 y x_{ip}. Para solucionar lo anterior se crea la variable auxiliar w_0^{ip} que trabaja como el producto de las variables en conflicto, el comportamiento deseado de esta variable es:

$$w_0^{ip} = \begin{cases} w_0 & \text{si } x_{ip} = 1 \\ 0 & \text{si } x_{ip} = 0 \end{cases} \tag{3.60}$$

Junto con la incorporación de la variable recientemente nombrada, es necesario agregar las siguientes restricciones para asegurar su correcto funcionamiento:

$$w_0^{ip} - \overline{T}x_{ip} \leq 0 \qquad \forall i \in I', p \in P \tag{3.61}$$

$$w_0^{ip} - w_0 \leq 0 \qquad \forall i \in I', p \in P \tag{3.62}$$

$$-w_0^{ip} + w_0 - \overline{T}\left(1 - x_{ip}\right) \leq 0 \qquad \forall i \in I', p \in P \tag{3.63}$$

La secuencia de resolución para el sub-problema se muestra en la Figura 3.1 y se detalla a continuación:

1. De la solución del maestro restringido se obtienen los valores de las variables duales y, a su vez, las decisiones de ramificación.

2. Se encuentra el valor óptimo de la F.O. sujeto a las restricciones que debe cumplir cada ruta factible, las restricciones necesarias para el correcto funcionamiento de w_0^{ip} y las restricciones de las decisiones de ramificación.

3. Si el valor encontrado es negativo se agrega la ruta al maestro restringido y se vuelve a iterar. De ser positivo y el valor de la expresión de η_{rt} también, entonces se calcula la diferencia entre ellos y si es negativo la ruta es agregada al maestro restringido y se vuelve a iterar. Finalmente si la diferencia anterior es positiva, concluye la generación de columnas.

El procedimiento evita agregar la condición ℓ en el sub-problema. Agregar dicha condición como restricción implicaría excluir soluciones factibles dado el comportamiento de η_{rt} (3.57), i.e. cuando el costo reducido es positivo y también es η_{rt} (debiendo la dual ser cero) pero la diferencia entre ambos es negativa. Evitando el problema de sobrestimar los costos reducidos al no atender el problema de CDR (Muter et al., 2013). Además, es necesario agregar las decisiones de branching como restricciones para evitar generar en el sub-problemas rutas que han sido fijadas en cero.

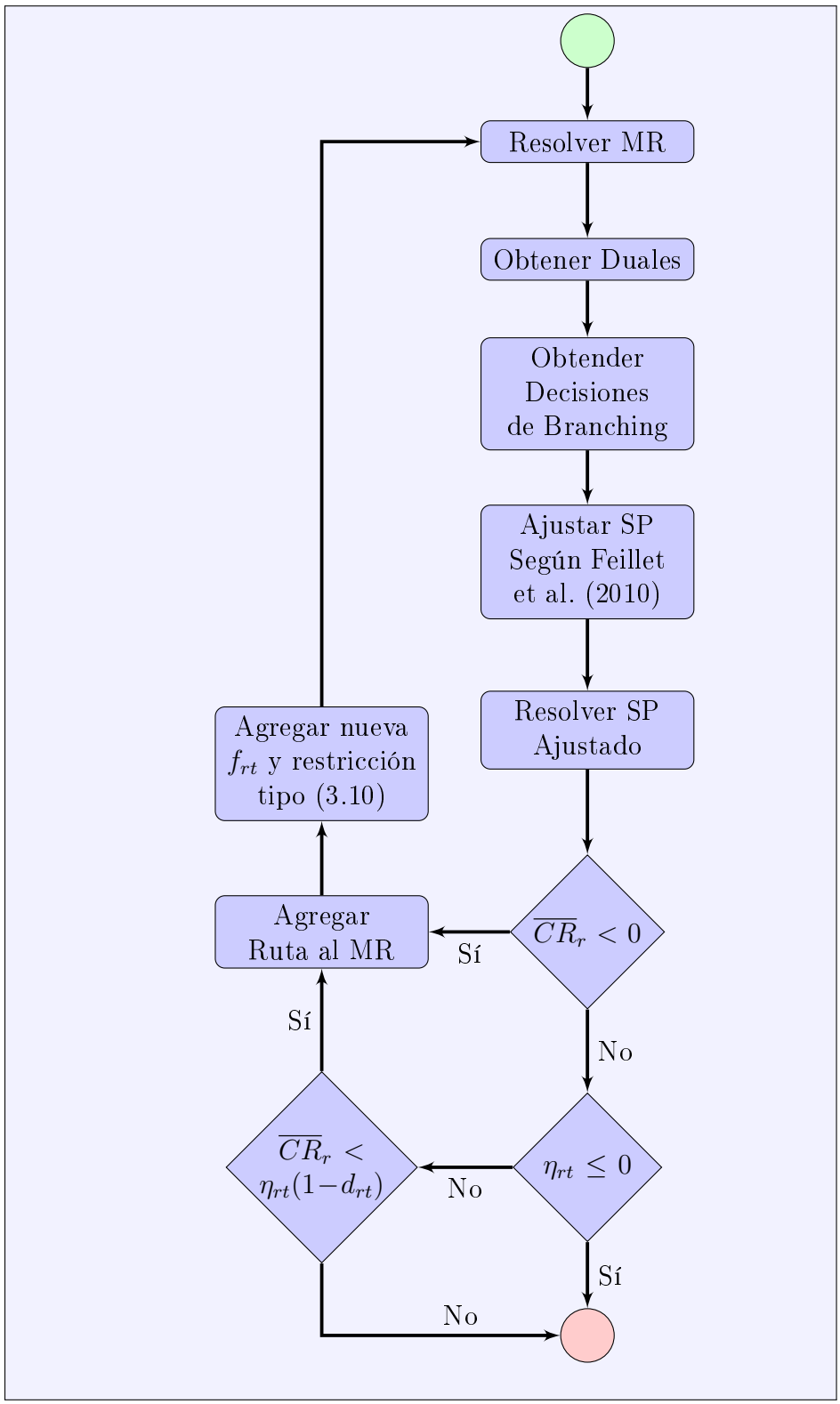


Figura 3.1: Procedimiento de solución del sub-problema

En la figura anterior se toma **MR** como maestro reducido, $\overline{CR}_r$ como costo reducido y **SP** como sub-problema.

3.3.4. Formulación del sub-problema

Finalmente el sub-problema de generación de columnas queda de la siguiente forma:

$$\begin{aligned}\text{mín} \quad & \sum_{ij\in A} c_{ij}y_{ij} + \sum_{i\in I',p\in P} \left[M_i\left(\alpha_{it}+\beta_{it}-\phi_{it}\right)+L_p\left(\phi_{it}+\zeta_{it}-\gamma_{it}\right)-\mu_{it}\right]x_{ip} \\ & + \sum_{i\in I'} \frac{\lambda_{it}}{\overline{T}}\left(\gamma_{it}+\delta_{it}-\phi_{it}-\zeta_{it}\right)\left(\sum_{p\in P} w_0^{ip} - w_i\right)\end{aligned} \tag{3.64}$$

Sujeto a

$$\sum_{j\in I} y_{ij} - \sum_{j\in I} y_{ji} = 0 \qquad \forall i \in I' \tag{3.65}$$

$$\sum_{j\in I'} y_{0j} = 1 \tag{3.66}$$

$$\sum_{i\in I'} y_{i0} = 1 \tag{3.67}$$

$$w_i - w_j + 2\overline{T}y_{ij} \leq 2\overline{T} - t_{ij} \qquad \forall i \in I', j \in I \wedge i \neq j \tag{3.68}$$

$$-w_j + t_{0j}y_{0j} \leq 0 \qquad \forall j \in I' \tag{3.69}$$

$$w_j - \overline{T}\sum_{i\in I'} y_{ij} \leq 0 \qquad \forall j \in I \wedge i \neq j \tag{3.70}$$

$$\sum_{ij\in A} t_{ij}y_{ij} \leq \overline{t} \tag{3.71}$$

$$\sum_{i\in I',p\in P} L_p x_{ip} \leq \overline{Q} \tag{3.72}$$

$$w_0 - \sum_{ij\in A} t_{ij}y_{ij} \leq 0 \tag{3.73}$$

$$w_0^{ip} - \overline{T}x_{ip} \leq 0 \qquad \forall i \in I', p \in P \tag{3.74}$$

$$w_0^{ip} - w_0 \leq 0 \qquad \forall i \in I', p \in P \tag{3.75}$$

$$-w_0^{ip} + w_0 - \overline{T}\left(1 - x_{ip}\right) \leq 0 \qquad \forall i \in I', p \in P \tag{3.76}$$

$$\text{Restricciones de Ramificación} \tag{3.77}$$

$$y_{ij} \in \{0,1\} \qquad \forall ij \in A \tag{3.78}$$

$$x_{ip} \in \{0,1\} \qquad \forall i \in I', p \in P \tag{3.79}$$

$$w_i \geq 0 \qquad \forall i \in I \tag{3.80}$$

$$w_0^{ip} \geq 0 \qquad \forall i \in I', p \in P \tag{3.81}$$

Donde (3.65), (3.66) y (3.67) se encargan de que todo nodo que sea visitado (distinto del depot) sea abandonado, que sólo hay una salida desde el depot y una sola llegada, respectivamente. Restricción (3.68) establece el momento en que el camión sale de cada cajero visitado tomando en cuenta el tiempo en que salió del cajero anterior y los tiempos de traslado y servicio (ambos contenidos en t_{ij} para j, por tanto, es el tiempo de salida). En (3.69) se estipula

el tiempo mínimo que debe tener el primer cajero visitado. (3.70) impide que la variable w_j pueda tener valor positivo cuando el cajero j no es visitado.

Las restricciones (3.71) y (3.72) son las restricciones de capacidad inherentes a cada ruta, la primera indica que el tiempo que demora en llegar el camión de vuelta al depot no puede exceder $\bar{t}$ unidades de tiempo, mientras que la segunda restringe la cantidad de efectivo que puede ser transportado en cada ruta, $\overline{Q}$.

La restricción (3.73) establece que el tiempo de llegada al depot no puede exceder la suma del tiempo de todos los arcos utilizados utilizados por la ruta seleccionada.

Las restricciones (3.74) a (3.76) son aquellas encargadas del correcto funcionamiento de w_0^{ip} como la expresión (3.60). Las restricciones (3.78) a (3.81) muestran la naturaleza de las variables presentes en el sub-problema.

El conjunto de restricciones en (3.77) vienen a imponer las decisiones de ramificación. Estas restricciones son necesarias cuando el algoritmo de ramificación fija alguna de las rutas presentes en el maestro reducido en cero, entonces es necesario imponer restricciones para que el sub-problema no vuelva a generar dicha ruta. Específicamente, para una ruta cualquiera fijada en cero, la restricción introducida al sub-problema establece que la suma de las variables binarias de uso de arcos y las de asignación de *cassette* a cajero será igual o menor a dos veces la cantidad de cajeros en la ruta. Lo anterior se puede ilustrar con la ruta ilustrada en la Figura 3.2. En la ruta se visitan 5 cajeros, en cada uno se dejan *cassette* tipo 1 o 2. La restricción que prohíbe la reproducción de tal ruta en el sub-problema es:

$$y_{D,2} + y_{2,3} + y_{3,7} + y_{7,6} + y_{6,9} + y_{9,D} + x_{2,1} + x_{3,1} + x_{7,2} + x_{6,1} + x_{9,2} \leq 10 \tag{3.82}$$

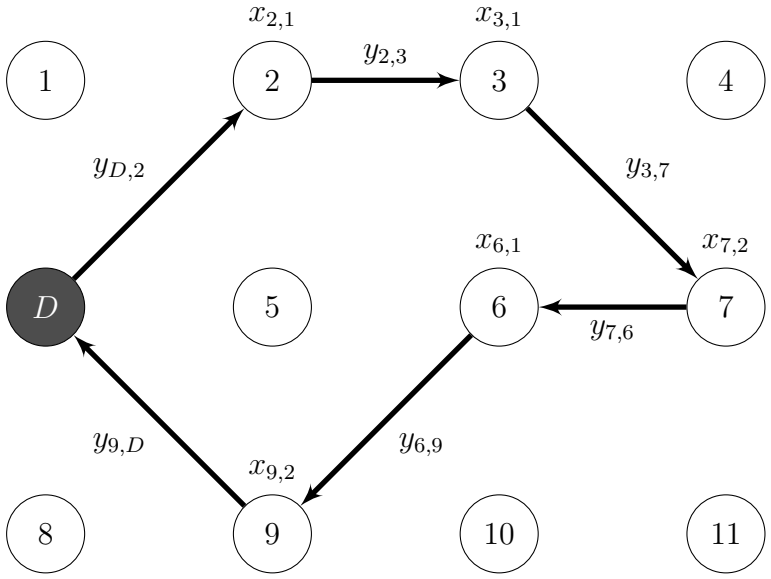


Figura 3.2: Ejemplo de ruta a excluir en sub-problema por decisión de ramificación

Con la restricción (3.82), la ruta de la Figura 3.2 no puede ser completamente reproducida. Puede seguirse el mismo recorrido pero al menos un *cassette* entregado debe ser diferente ya

que de lo contrario se estaría violando la restricción. Si bien se puede invertir el orden o intercambiar un par de nodos visitados (por ejemplo ir de 3 a 6 y de 6 a 7), la ruta no seria equivalente puesto que los tiempos de visita a estos cajeros sería diferente y los parámetros de demanda remanente post visita no serían iguales.

Una vez construido el sub-problema de generación de columnas, la resolución de este se hace como un MIP. Si se encuentra una ruta que viole la restricción dual (3.33), deben construirse los parámetros Q_{irt}, $\overline{\Lambda}_{irt}$, d_{rt} y c_{rt} antes de agregar la variable x_{rt} correspondiente al maestro restringido. Al agregar dicha variable se debe agregar f_{rt} para habilitar la salida retrasada de la ruta y la restricción tipo (3.10) asociada a tales variables para evitar que la suma del retraso en la salida de la ruta más el tiempo que toma en completarse sea mayor a la extensión del período.

3.4. Estrategia de Branching y Selección de Nodos

En esta sección se indican y describen las estrategias de branching y selección de nodos utilizadas en el algoritmo de *Branch & Price*. También, al final de la sección, se describen las soluciones iniciales provistas al algoritmo y la heurística empleada en caso de infactibilidad del maestro restringido.

3.4.1. Estrategia de Branching

La estrategia de branching utilizada es la descrita en Achterberg et al. (2005), *Reliability Branching*. En esta se utiliza *Strong Branching* y *Pseudocost Branching*, utilizando lo mejor de cada estrategia. *Strong branching* es utilizado en nodos no inicializados, o en nodos inicializados cuyo pseudocosto no se considera confiable.

Considerando el maestro restringido, tomamos un sub-problema de un nodo no podado, su valor óptimo z_Q y solución $\overline{x}_Q$. Sea S el set de índices para las variables $\overline{x}_i \in \overline{x}_Q$, y el set $K = \{i \in S | \overline{x}_i \notin \{1, 0\}\}$ de variables candidatas a ser escogidas para branching, es decir, fraccionales. Para las cuales se define $f_i^+ = \lceil \overline{x_i} \rceil - \overline{x_i}$ y $f_i^- = \overline{x_i} - \lfloor \overline{x_i} \rfloor$.

Para cada $i \in K$ se establecen Q_i^+ y Q_i^- como los dos sub-problemas de branching, derecho e izquierdo, en donde se agregan las restricciones $x_i \geq \lceil \overline{x}_i \rceil$ y $x_i \leq \lfloor \overline{x}_i \rfloor$, respectivamente. El cambio en la función objetivo de estos será $\Delta_i^+ = z_{Q_i^+} - z_Q$ y $\Delta_i^- = z_{Q_i^-} - z_Q$, estos valores son puestos en una sola función score para ser comparados:

$$score\left(q^-, q^+\right) = (1-\mu)\,\text{mín}\{q^-, q^+\} - \mu\,\text{máx}\{q^-, q^+\} \tag{3.83}$$

Donde $\mu \in [0, 1]$[1]. Este tipo de función score es típicamente utilizada para estos fines (Linderoth and Savelsbergh, 1999).

[1]El valor utilizado en SCIP es de $\mu = 1/6$

En *pseudocost branching*, regla inicialmente propuesta por Bénichou et al. (1971), se utilizan los cambios en la función objetivo por unidad de cambio en la variable ramificada para computar los pseudocostos:

$$\varsigma_i^+ = \frac{\Delta_i^+}{f_i^+} \qquad \text{y} \qquad \varsigma_i^- = \frac{\Delta_i^-}{f_i^-} \tag{3.84}$$

Con esto se define σ_i^+ como la suma de de los ς_i^+ sobre todos los problemas en Q, donde la variable i ha sido seleccionada como variable de branching y el problema Q_i^+ es factible y ha sido resuelto. Adicionalmente η_i^+ es el número de estos problemas. Finalmente los pseudocostos serán:

$$\Psi_i^+ = \frac{\sigma_i^+}{\eta_i^+} \qquad \text{y} \qquad \Psi_i^- = \frac{\sigma_i^-}{\eta_i^-} \tag{3.85}$$

Para realizar comparaciones entre candidatos se utiliza la siguiente función de score:

$$s_i = score\left(f_i^- \Psi_i^-, f_i^+ \Psi_i^+\right) \tag{3.86}$$

El principal problema en esta estrategia de branching es que al comienzo del algoritmo no hay información con la cual calcular esos valores, teniendo $\sigma_i^+ = \eta_i^+ = 0$. Para variables en tal estado, se dice que los pseudocostos de estas variables, $i \in S$, están no inicializados en dirección superior cuando $\eta_i^+ = 0$. En este caso sus pseudocostos se fijan en $\Psi_i^+ = \Psi_{avg}^+$ donde Ψ_{avg}^+ es el promedio de los pseudocostos de las variables inicializadas en la misma dirección. En caso de que ninguna variables se encuentre inicializada este parámetro es igual a 1. En consecuencia, en el inicio del algoritmo este toma decisiones en la parte superior del árbol con pseudocostos en los que no se puede confiar, esto es un problema puesto que justamente en la parte superior del árbol es en donde las decisiones de branching tienen el mayor impacto en la estructura del mismo (Achterberg et al., 2005).

Por otro lado *strong branching*, propuesto inicialmente en Applegate et al. (1995) en el contexto del problema del vendedor viajero, trata de evaluar cuál de las variables fraccionales entrega el mejor progreso de la cota inferior antes de tomar la decisión de branching. Esta evaluación es realizada introduciendo temporalmente las restricciones $x_i \geq \lceil x_i \rceil$ y, subsecuentemente, $x_i \leq \lfloor x_i \rfloor$ para la variable x_i de valor $\overline{x}_i \notin \mathbb{Z}$ en el LP, para luego resolver tales sub-problemas.

Si el set completo $K = \{i \in S | \overline{x}_i \notin \{1, 0\}\}$ es considerado como candidato y se solucionan los subsecuentes sub-problemas a optimalidad, la estrategia se pasa a llamar *full strong branching*. Aunque en general esta regla tiene buenos resultados con respecto al número de nodos necesarios para resolver problemas completos, realizar este procedimiento en cada uno de los nodos es computacionalmente costoso. Por consiguiente, existen opciones para acelerar esta estrategia, la primera de ellas es restringir el set de variables candidatas de alguna forma, por ejemplo, considerar sólo las binarias fraccionales más cercanas a cero (debido a que en la solución final la mayor parte de ellas tomarán dicho valor). Una segunda opción es limitar el número de iteraciones que el algoritmo de simplex (o dual simplex, según sea el caso) realiza en cada sub-problema, ya que usualmente la magnitud del cambio en la función objetivo decrece con cada iteración realizada (rendimientos decrecientes). En tal caso, de utilizar ambas

aceleraciones, los parámetros que debiesen especificarse son: k el máximo número de candidatos a evaluar, γ el número máximo de iteraciones a realizar por el algoritmo de simplex y el criterio según el cuál se seleccionan los candidatos a evaluar.

Algorithm 4: Reliability Branching

Input: Actual sub-problema, Q, con solución lineal óptima $\overline{x} \in X_{MIP}$.
Result: Índice $j \in S$ de la variable x_j con valor fraccional $\overline{x}_j \notin \mathbb{Z}$.

Sea $H = \{i \in S | \overline{x}_i \notin \mathbb{Z}\}$ el set de candidatos, tal que $H \neq \emptyset$.
foreach $i \in H$ **do**
  $f_i^+ \leftarrow \lceil \overline{x}_i \rceil - \overline{x}_i$
  $f_i^- \leftarrow \overline{x}_i - \lfloor \overline{x}_i \rfloor$
  Actualizar η_i^- y η_i^-
  foreach $k \in \{-, +\}$ **do**
    if $\eta_i^k = 0$ **then**
      $\Psi_i^k = \Psi_{avg}^k$
    else
      $\varsigma_i^k = \dfrac{\Delta_i^k}{f_i^k}$
      $\Psi_i^k = \dfrac{\sigma_i^k}{\eta_i^k}$
    end
  end
  $s_i = score\left(f_i^- \Psi_i^-, f_i^+ \Psi_i^+\right)$
end
Ordenar las variables en orden no creciente con respecto a sus pseudocostos.
Se fija $\vartheta = 0$ y $\overline{s} = \text{máx}_{i \in H}\{s_i\}$.
for $i \in H' = \{i \in H | \text{mín}\{\eta_i^-, \eta_i^+\} < \eta_{rel}\}$ **do**
  Realizar a lo más γ iteraciones de simplex (o dual simplex) en los sub-problemas Q_i^+ y Q_i^-.
  Obtener las correspondientes ganancias en valor objetivo $\widehat{\Delta}_i^+$ y $\widehat{\Delta}_i^-$.
  Actualizar los valores Ψ_i^+ y Ψ_i^- y s_i.
  if $\overline{s} \leq s_i$ **then**
    $\overline{s} = s_i$
    $\vartheta = 0$
  else
    $\vartheta = \vartheta + 1$
    if $\vartheta = \lambda$ **then**
      stop
    end
  end
end
return $j \leftarrow \text{argmax}_{i \in H}\{s_i\}$

Reliability branching une las dos reglas anteriores tratando de evitar los problemas de cada una ellas. La regla indica que para variables con pseudocostos no inicializados o con pseudocostos no confiables se utilizará *strong branching*, mientras que para el resto será *pseudocost*

branching. Específicamente, se considerará que los pseudocostos de cualquier variable no son confiables si se cumple que mín$\{\eta_i^-, \eta_i^+\} < \eta_{rel}$, siendo η_{rel} el parámetro de confiabilidad. Todo lo anterior es detallado en el algoritmo 4, describiendo paso a paso una iteración en que la regla determina sobre qué candidato ramificar. En la implementación de esta tesis el parámetro de confiabilidad tomará un valor de 3.

3.4.2. Estrategia de Selección de Nodos

Luego de que un sub-problema ha sido procesado, el esquema de solución continua con un sub-problema que sea un hoja del actual árbol de *B&B*. Esta selección tiene dos propósitos que son vistos como opuestos (Achterberg, 2007):

1. Encontrar una solución factible que mejore el *upper bound*, lo que ayuda en la poda del árbol de ramificación.

2. Mejorar el *lower bound* global del esquema.

Depth First trata seleccionar uno de los nodos hijos del actual sub-problema, si es podado la estrategia hace un *backtrack* al primer antecesor que tenga nodos descendientes no procesados, es decir, siempre trata de escoger un nodo de la ramificación más profunda del árbol. Esta estrategia de selección de nodos tiene dos ventajas, la primera de ellas es que encuentra soluciones factibles (no necesariamente buenas) de forma rápida y la segunda es que los sub-problemas seleccionados usualmente son muy similares al que actualmente procesado, por tanto la carga del manejo de los sub-problemas es mínimo.

Por otro lado, *Best First* se enfoca en mejorar la cota inferior global lo más rápido posible al siempre seleccionar el sub-problema que presenta la menor cota inferior de entre todas las hojas del árbol, sin embargo sus soluciones suelen estar lejos de tener variables con valores enteros. Esta estrategia tiene la ventaja de procesar la menor cantidad de nodos, dada una estrategia de ramificación fija.

Best Estimate Search, estrategia seleccionada, apunta a tener lo mejor de las dos anteriores. Busca encontrar buenas soluciones de forma rápido. Esta regla calcula el valor estimado e_Q para los sub-problemas factibles que puedan estar contenidos en el sub-árbol representado por las hojas de Q, se selecciona el nodo con el mínimo valor estimado. Este valor estimado combina información tanto de la cota inferior como de la integralidad de la solución de la relajación lineal actual.

Para el cálculo de e_Q se utiliza la expresión de Forrest et al. (1974) utilizada en su *best estimate rule*. Esta expresión hace uso de los pseudocostos de las variables del problema para estimar el cambio en la función objetivo. La expresión es:

$$e_Q = \check{c}_Q + \sum_{i \in S} \text{mín}\{f_i^- \Psi_i^-, f_i^+ \Psi_i^+\} \tag{3.87}$$

Donde $\check{e}_Q$ es la cota inferior del actual sub-problema.

3.4.3. Farkas Pricing

La idea de *Farkas Pricing* es, a través del Lemma de Farkas, encontrar una variables que, en caso de infactibilidad del maestro restringido, le devuelva la factibilidad al problema o, en caso contrario, probar la infactibilidad del nodo y podarlo.

El lemma de Farkas dice, dados los problemas primal y dual siguientes,

$$\begin{aligned} \text{mín} \quad & c_N x_N && (3.88) \\ & s.a. && \\ A._{,N}\, x_N & \geq a && (3.89) \\ B._{,N}\, x_N & = b && (3.90) \\ x_N & \in \mathbb{N} \cup \{0\} && (3.91) \end{aligned} \qquad \begin{aligned} \text{máx} \quad & a^T\pi + b^T\sigma && (3.92) \\ & s.a && \\ A^T\pi + B^T\sigma & \leq c && (3.93) \\ \pi & \geq 0 && (3.94) \\ \sigma & \in \mathbb{Q} && (3.95) \end{aligned}$$

Lemma 3.4.1 (Farkas)
Para el modelo primal cuya F.O. es 3.88, exactamente una de las siguientes alternativas es cierta:

i. $\exists x_N \in \mathbb{N} \cup \{0\}$ tal que $A._{,N}\, x_N \geq a$ y $B._{,N}\, x_N = b$.

ii. $\exists \mu \geq 0, \nu \in \mathbb{Q}$ tal que $\mu^T A._{,N} + \nu^T B._{,N} \leq 0^T$ y $\mu^T a + \nu^T b > 0$.

La prueba del lemma 3.4.1 se puede encontrar en Bertsimas and Tsitsiklis (1998). Con este lemma podemos inferir que dado un RMP infactible, existe un vector (μ, ν) que representa un rayo no acotado en el dual tal que cumple la condición (ii) del lemma 3.4.1. Por consiguiente la idea de este sub-problema de generación de columnas es encontrar una variable primal x_i tal que su restricción dual corte tal rayo, en caso contrario se prueba la infactibilidad del maestro relajado y se procede a podar el nodo actual.

El agregar el vector (μ, ν), que demuestra que la condición (ii) del lemma 3.4.1 se cumple, a una solución dual factible no afecta de forma alguna factibilidad de la solución puesto que $\mu^T A._{,N} + \nu^T B._{,N} \leq 0^T \leq c^T$, pero si modifica el valor objetivo de dicha solución dual. En lo que sigue, vectores μ y ν son llamados *multiplicadores de Farkas*.

Para contradecir, o corromper, la condición (ii) del lemma 3.4.1 debemos encontrar una variable primal x_i tal que:

$$\mu^T A._{,i} + \nu^T B._{,i} > 0 \tag{3.96}$$

Esta variable x_i induce la creación de una restricción tipo (3.93), la cual es siempre violada cuando se agrega un múltiplo lo suficientemente grande de los multiplicadores de Farkas anteriormente mencionados. En consecuencia, el rayo (μ, ν) ya no es factible para el dual del RMP que contiene x_i. Este sub-problema de generación de columnas nos conlleva a encontrar el siguiente resultado:

$$\hat{c} = \text{mín}\{-A^T_{.,i}\mu - B^T_{.,i}\nu\} \tag{3.97}$$

Como se puede observar el problema de *Farkas pricing* es similar al sub-problema de generación de columnas visto en la sección 3.3.4, sólo se deben hacer cero los parámetros $c_{ij} \in A$ y ocupar los multiplicadores de Farkas en vez de los valores duales del RMP infactible que denotaremos de igual forma pero con un sombrero encima (ej. $\hat{\alpha}_{it}$).

La implementación de la idea anterior, *Farkas pricing*, es bastante directa, una vez detectada la infactibilidad de un maestro relajado se obtienen los multiplicadores de Farkas (entregados por el solver) y para cada período resolvemos:

$$\hat{c}_{rt} = \text{mín} \left\{ \begin{array}{l} 0 + \displaystyle\sum_{i \in I', p \in P} \left[M_i \left(\hat{\alpha}_{it} + \hat{\beta}_{it} - \hat{\phi}_{it} \right) + L_p \left(\hat{\phi}_{it} - \hat{\zeta}_{it} - \hat{\gamma}_{it} \right) - \hat{\mu}_{it} \right] x_{it} \\ + \displaystyle\sum_{i \in I'} \frac{\lambda_{it}}{\overline{T}} \left(\hat{\gamma}_{it} + \hat{\delta}_{it} - \hat{\phi}_{it} - \hat{\zeta}_{it} \right) \left(\displaystyle\sum_{p \in P} w_0^{ip} - w_i \right) \end{array} \right\} \tag{3.98}$$

Sujeto a restricciones (3.65) a (3.81)

Si en alguno de los sub-problemas encontramos una solución en la cual $\hat{c}_{rt} > 0$, entonces agregamos la ruta asociada a tal solución al maestro relajado, de no encontrar ningún período en donde su solución cumpla con dicha condición se prueba la infactibilidad del maestro relajado y el nodo asociado es podado.

3.4.4. Rutas iniciales

Para evitar que el problema maestro sea infactible al comienzo de la resolución del problema, a este se le entrega un set Ω de rutas, que asegura la factibilidad de este. Dicho set esta compuesto de dos subsets: Ω_S y Ω_{BP}. Dentro del primer set (Ω_S) se encuentran rutas simples que visitan sólo un cajero y entregan el *cassette* de mayor valor, tal como se detalla en el algoritmo 5. El tamaño de dicho set será igual al producto del número de cajeros y períodos en la instancia.

Algorithm 5: Construcción de Rutas Iniciales

Input: Número $|I'|$ de cajeros, $|T|$ de períodos, el valor q del *cassette* de mayor valor y los parámetros $c_{ij}, t_{ij}\ \forall (ij) \in A$, $\overline{T}$ y λ_{it}.

Result: Set Ω_S de rutas iniciales y sus respectivos parámetros $Q_{irt}, \overline{\Lambda}_{irt}, c_{rt}, d_{rt}$.

```
for i ∈ I' do
    c_rt = c_0i + c_i0
    d_rt = (t_0i + t_i0) / T̄
    Q_irt = q
    for t ∈ T do
        Λ̄_irt = (1 − d_rt) λ_it
        Ω_S = Ω_S ∪ {[c_rt, d_rt, Q_irt, Λ̄_irt]}
    end
end
```

El subset Ω_{BP} esta conformado por rutas generadas por un algoritmo de barrido polar, similar al presentado en Gillett and Miller (1974), clásico algoritmo de agrupar y rutear. Este se describe en detalle en algoritmo 6. Dentro de las particularidades de esta implementación está el hecho de que se tomarán en cuenta no más de 5 visitas en cada ruta, con el fin de no violar las restricciones de tiempo y carga impuestas sobre las rutas. Cada cajero tiene asociado un par de coordenadas cartesianas. Al transformar estas coordenadas cartesianas a coordenadas polares se tomará el depot como el punto de origen, haciendo los ajustes correspondientes en cada cajero. Los problemas de vendedor viajero (TSP, por sus siglas en inglés) se resuelven con una programación dinámica simple. Finalmente en cada vista sólo se entregará el *cassette* de menor valor, nuevamente, con el fin de no lidiar con la restricción de capacidad.

En esta implementación, al igual que en el trabajo original, se recorren, de forma ordenada, todas las coordenadas polares (cajeros) de forma creciente y decreciente. Dado lo anterior, tendremos dos sets de rutas: las rutas obtenidas de recorrer la lista de cajeros de forma creciente y decreciente. Luego de agrupar los cajeros en clusters (set de cajeros que conforman el conjunto $\mathbb{A}$ del algoritmo 6), se chequea que ninguno de estos tenga sólo un cajero, puesto que dichas rutas son generadas por el algoritmo 5. Estos clusters son el input de los TSP que generarán las rutas finales del algoritmo. Se permite que existan clusters que posean miembros idénticos, suceso que puede darse al recorrer las listas en ambos sentidos. Como los costos de viaje son simétricos, el TSP va a generar dos rutas diferentes de igual costo: Una en orden creciente según coordenadas polares y otra de forma decreciente según las mismas coordenadas, sin embargo para el problema maestro estas no son iguales puesto que existen costos asociados al inventario de cada cajero y, como ya sabemos, el momento de llegada a cada cajero importa, en consecuencia los inventarios finales de los cajeros de dichas rutas serán diferentes.

Finalmente, en cada período se incorporan todas las rutas generadas por el algoritmo de barrido polar. Para ello se deben construir los parámetros correspondientes para dichas rutas tal como se detalla en el algoritmo 6.

Algorithm 6: Construcción de Rutas Iniciales - Barrido Polar

Input: Número $|I'|$ de cajeros, $|T|$ de períodos, el valor q del *cassette* de menor valor, las coordenadas cartesianas de todos los cajeros y el depot (x_j, y_j) y los parámetros $c_{ij}, t_{ij}\ \forall (ij) \in A$, $\overline{T}$ y λ_{it}.

Result: Set Ω_{BP} de rutas iniciales y sus respectivos parámetros $Q_{irt}, \overline{\Lambda}_{irt}, c_{rt}, d_{rt}$.

for $i \in I'$ **do**
  Para cada cajero, obtengo sus coordenadas polares (ρ_i, θ_i):
  $\rho_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$
  $\theta_{rt} = \tan^{-1}\left(\frac{y_i - y_0}{x_i - x_0}\right)(180/\pi)$
end
Se generan 2 vectores, v_c y v_d, con los cajeros ordenados de forma creciente y decreciente en θ_i, con segunda prioridad de orden ρ_i, respectivamente.
Se crea un conjunto $\mathbb{A}$ vacío que servirá como conjunto auxiliar en la creación de las rutas y el conjunto $\mathbb{B}$ que agrupará las rutas creadas.
En la generación de rutas a partir de de los vectores ordenados:
foreach $\mathbb{V} \in \{v_c, v_d\}$ **do**
  $\mathbb{A} = \emptyset$
  for $n \in \mathbb{V}$ **do**
    if $|\mathbb{A}| < 4 \wedge n < \|\mathbb{V}\|$ **then**
      $\mathbb{A} = \mathbb{A} \cup n$
    else
      if $|\mathbb{A}| = 4 \vee n = \|\mathbb{V}\|$ **then**
        $\mathbb{A} = \mathbb{A} \cup n$
        if $|\mathbb{A}| > 1$ **then**
          $\mathbb{B} = \mathbb{B} \cup TSP\,(\mathbb{A}, \{c_{ij} | i, j \in I\})$ ← Se resuelve TSP para conjunto $\mathbb{A}$.
        end
        $\mathbb{A} = \emptyset$
      end
    end
  end
end
En cada período se incluyen las rutas agrupadas en $\mathbb{B}$:
for $t \in T$ **do**
  foreach $r \in \mathbb{B}$ **do**
    Se generan los parámetros $\{[c_{rt}, d_{rt}, Q_{irt}, \overline{\Lambda}_{irt}]\}$ para ruta r.
    $\Omega_{BP} = \Omega_{BP} \cup \{[c_{rt}, d_{rt}, Q_{irt}, \overline{\Lambda}_{irt}]\}$
  end
end

Finalmente la figura 3.3 muestra el diagrama de flujo de la resolución del *Branch & Price* planteado.

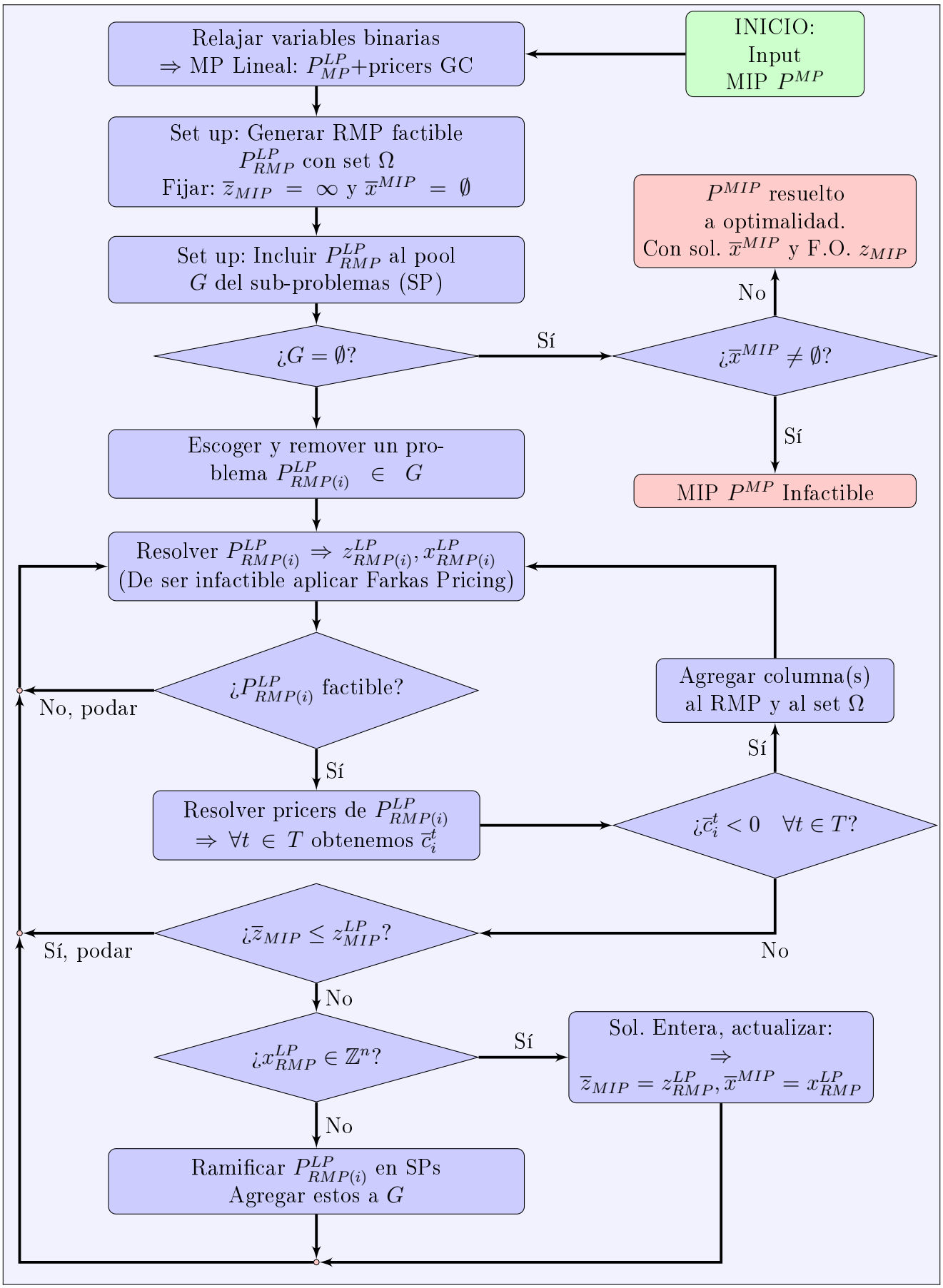


Figura 3.3: Diagrama de Flujo *Branch & Price*

Capítulo 4

Resultados

En este capítulo se detallan y recopilan los experimentos realizados y los resultados obtenidos. En primer lugar se establecen los valores de los parámetros existentes en el problema y se muestra la red sobre la que el algoritmo es empleado. Segundo, al ver los primeros resultados se evalúa un set de mejoras al algoritmo y su impacto en la resolución del problema. En tercer lugar se crean 3 escenarios de demanda con el fin de modificar las soluciones entregadas por el algoritmo y evaluar su comportamiento, poniendo énfasis en el uso de las particularidades planteadas en el capítulo 2, de definición del problema.

4.1. Parámetros y escenario inicial

En esta sección se describe la red de cajeros objetivo y los parámetros que acompañan el escenario inicial a evaluar. Los datos disponibles contienen un conjunto de 100 cajeros automáticos distribuidos en un plano cartesiano, como muestra la figura 4.1. Se toma la distancia euclidiana como la distancia entre nodos, tal como indica la expresión (4.1), con un costo fijo de 6 unidades pecuniarias por kilómetro recorrido y el tiempo que demora un camión en recorrer las distancias entre cajeros se calcula utilizando una velocidad fija de 20 kilómetros por hora.

$$d_E\left(N_1, N_2\right) = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2} \tag{4.1}$$

El set de datos también contiene los inventarios iniciales y las demandas de cada cajero para hasta 42 periodos. En compañían de lo anterior están los valores de los dos tamaños de *cassette* disponibles para ser entregados, cuyos valores son de 40 y 65 unidades pecuniarias. Parte de las condiciones del problema indican que el tamaño del *cassette* más pequeño debe cubrir la demanda de un periodo de cualquier cajero en todo periodo, cosa que queda confirmada en la tabla 4.1, que entrega estadísticas descriptivas de las demandas contenidas en los datos disponibles.

Tabla 4.1: Estadísticas básicas de demandas por cajero - periodo

	Mín.	1er Qu.	Mediana	Media	3er Qu.	Máx.
demanda	0.00	0.85	1.86	2.31	3.41	9.06

Los costos de incumplir en la demanda por efectivo en los cajeros, como ya fue explicado, son dos: El costo fijo, con un valor de 30 unidades pecuniarias por caer en quiebre de stock, y el costo variable que es de 2 unidades pecuniarias por cada unidad de demanda no cumplida. Mientras que el costo de mantener inventario en dichos cajeros es de 0.5 unidades pecuniarias por unidad de inventario.

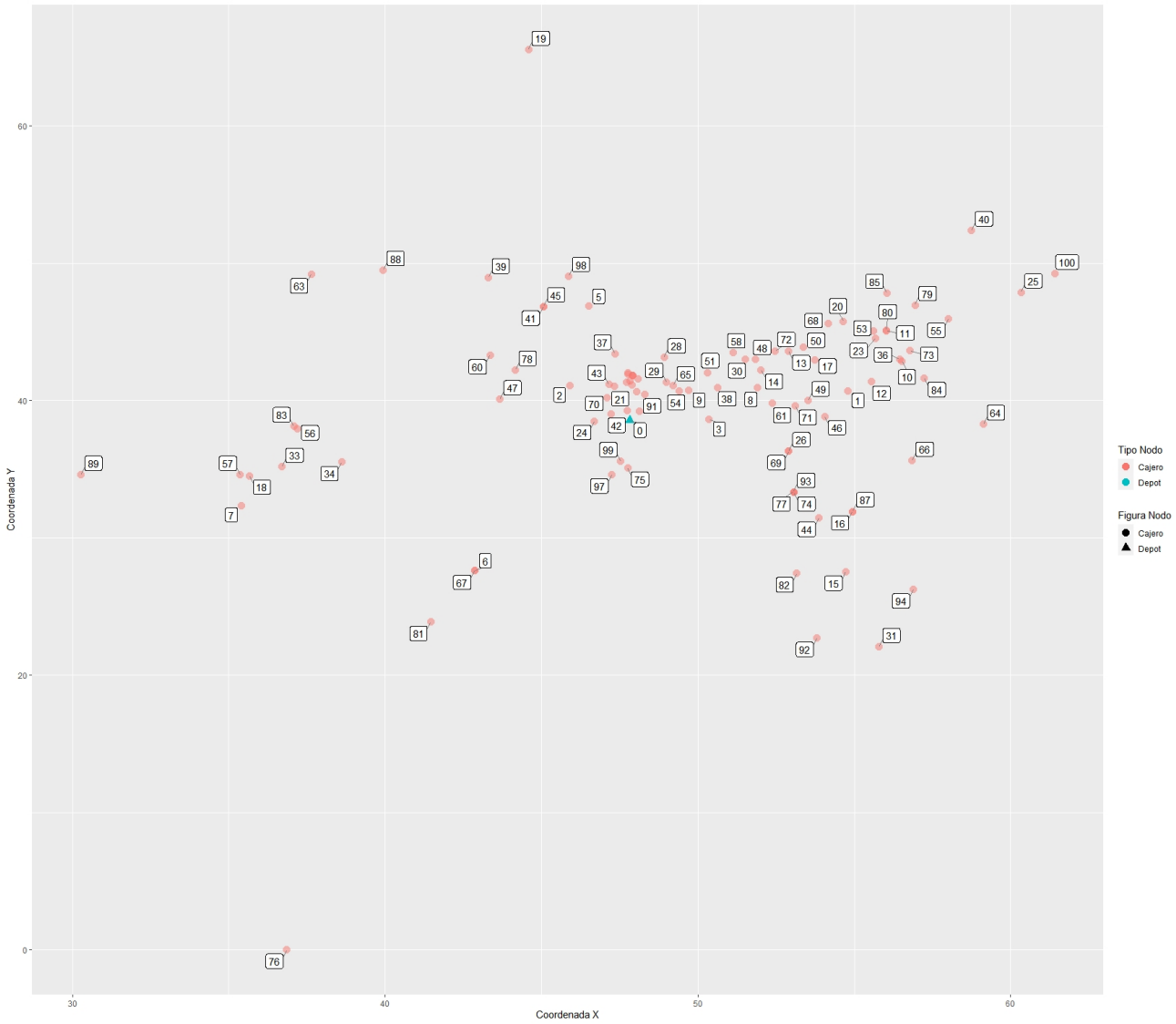


Figura 4.1: Disposición de cajeros y depot de la red

Finalmente, cada periodo tiene un largo de 8 horas. Las rutas que se generen durante la resolución del problema deben respetar un máximo de 4 horas de largo y su carga no puede superar las 300 unidades pecuniarias.

4.2. Pruebas iniciales

4.2.1. Primera prueba

La primera prueba se hizo con un conjunto de 3 cajeros y tomando en cuenta un horizonte de tiempo de 3 periodos. Las demandas para dichos cajeros y su inventario inicial se encuentran en la tabla 4.2. Esta prueba se realizó antes de ingresar las restricciones (3.11) a (3.13), que fueron incorporadas en la versión posterior.

Tabla 4.2: Inventario inicial y demandas para cajeros de primera prueba

Cajero	Inventario Inicial	Periodo		
		T1	T2	T3
C1	5.0	4.01	3.34	0.97
C2	5.0	7.93	5.28	1.53
C3	1.0	1.62	1.22	0.30

Los resultados de esta prueba se observan en la tabla 4.3, en la que se puede observar un gap en el nodo raíz considerable, indicando que la relajación lineal del problema maestro tiene deficiencias. El tiempo de resolución para esta instancia alcanza los 56 minutos y 36 segundos.

Tabla 4.3: Resultados primera prueba

Tiempo Resolución (s)	3396
Gap Nodo Raíz	843.5 %
Función Objetivo	272.0719
Número Nodos	8941
Nodos Internos	4483
Nodos Singulares	4458
Profundidad Máxima	108
Número Rutas Creadas	184

De hecho, la convergencia de las cotas inferior y superior es sumamente lenta, siendo que el óptimo es encontrado en un tercio del tiempo que toma la instancia en ser resuelta, tal como se puede observar en la figura 4.2. Del total de rutas generadas el óptimo sólo utiliza una que visita los 3 cajeros en el primer periodo. Dicha ruta, tal como se puede ver en la tabla 4.4, se ejecuta en el primer periodo, sin demora alguna en la salida, entrega en cada cajero el cassette de menor valor.

La tabla 4.4 nos muestra las variables de inventario para los 3 cajeros a través de los periodos del horizontes de tiempo de la prueba, en donde la segunda y tercera columna muestran justamente el cajero y periodo de la variable de inventario observada, con los índices empezando en uno.

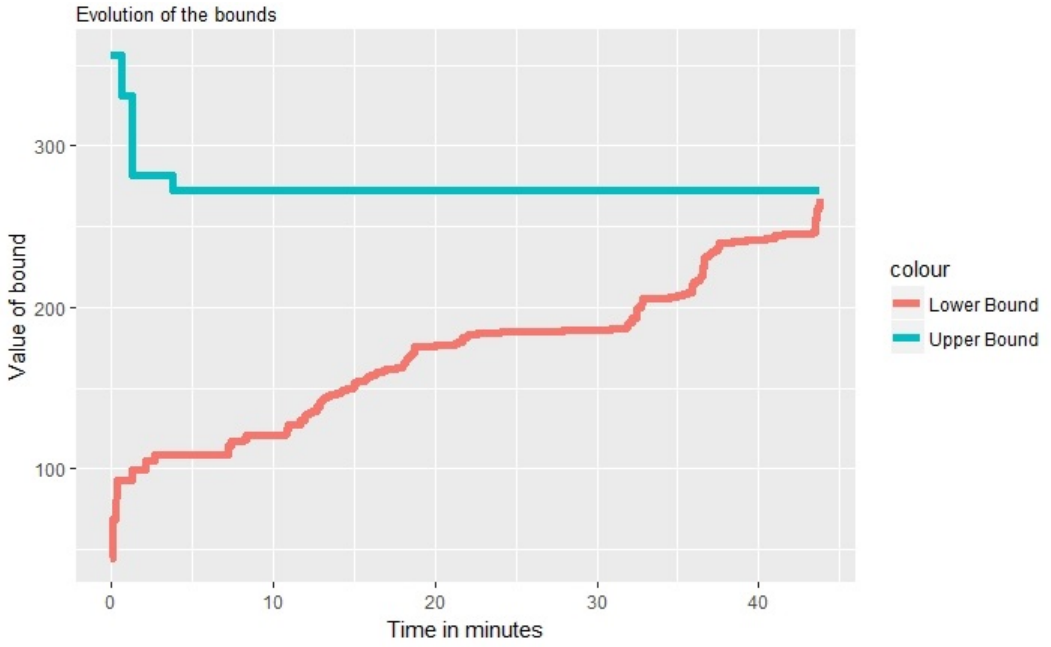


Figura 4.2: Convergencia de cotas en instancia inicial

En la primera columna de la tabla 4.4 se observa el inventario inicial de cada cajero en dicho periodo (notar que en el caso del primer periodo de cada cajero se muestra el inventario inicial), mientras que en la cuarta esta el inventario final del cajero. En la quinta se encuentran las demandas de los cajeros, en la sexta se muestra el valor del *cassette* entregado en dicho cajero si es que en tal periodo hay alguna ruta que los visite y en la séptima encontramos el segundo (de los 28800 existentes en cada periodo) en que la visita se realiza. Como se puede observar, la ruta utilizada visita los 3 cajeros en el primer periodo, dentro de los primeros 10 minutos.

Tabla 4.4: Inventarios de la solución de la primera prueba

Inv. Inicial	Cajero	Periodo	Inv. Final	Demanda	Cassette	T. Llegada
5.00	C1	T1	36.29	4.01	40.0	2178
36.29	C1	T2	32.95	3.34	-	-
32.95	C1	T3	31.98	0.97	-	-
5.00	C2	T1	32.23	7.93	40.0	576
32.23	C2	T2	26.95	5.28	-	-
26.95	C2	T3	25.42	1.53	-	-
1.00	C3	T1	0.00	1.62	40.0	3060
38.55	C3	T2	37.33	1.22	-	-
37.33	C3	T3	37.03	0.30	-	-

La figura 4.3 muestra el comportamiento del inventario en el tercer cajero a través de los 3 periodos, se observa la visita poco antes de cumplir una hora de ruta (minuto 51 para ser exactos), entregando un cassete pequeño, de 40 unidades pecuniarias.

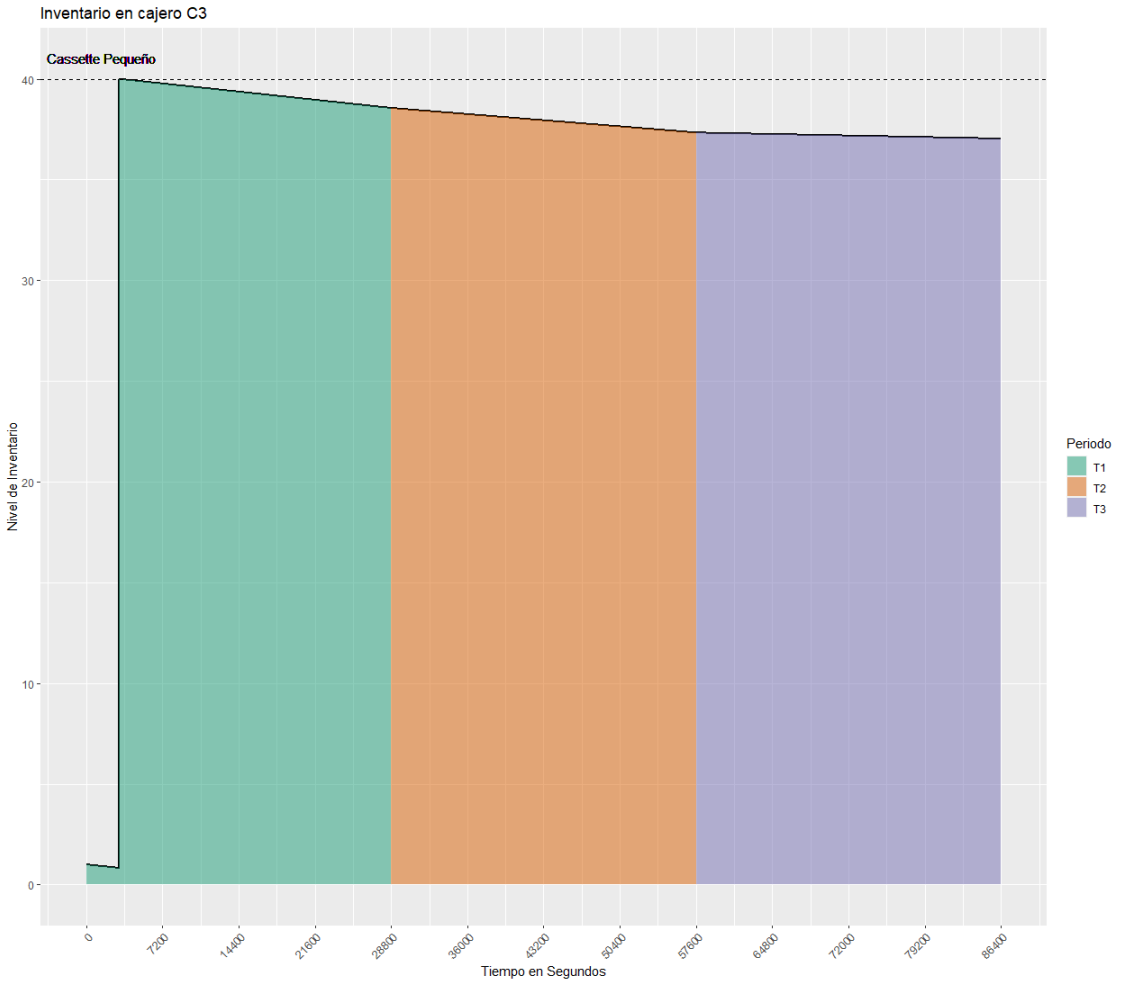


Figura 4.3: Evolución del inventario en el cajero 3 en prueba inicial

4.2.2. Inclusión de reglas sobre el nivel de servicio

Con el fin de mejorar la relajación lineal del problema maestro, disminuir el tiempo de resolución e introducir reglas de negocio relevantes para el problema, se introducen las restricciones (3.11) a (3.13) al maestro, estas indican lo siguiente:

- En ningún periodo el 50 % o más de los cajeros puede caer en quiebre de stock.
- Ningún cajero puede caer en quiebre de stock en un 50 % o más del total de periodos del horizonte de tiempo.
- Del total de combinaciones *cajero-periodo* del problema, no más del 25 % puede tener quiebre de inventario.

La idea de utilizar estas restricciones proviene de que indirectamente afecta el uso de las variables de rutas (fuerzan su uso al tratar de evitar quiebres de stock) y además, como no hay uso de variables de ruta en ellas, no es necesario modificar el subproblema de generación de columnas.

La tabla 4.5 resume los resultados de estas incorporaciones en la misma instancia de la prueba inicial, donde podemos ver que el tiempo de resolución de la instancia cae en un 45.9 % mientras que el tamaño del árbol (mirando como el número total de nodos en este) se reduce un 24.7 % (2206 nodos menos). La solución es idéntica a la encontrada en la prueba inicial, una ruta visita los 3 cajeros en el primer periodo, en el mismo orden y los mismos tiempos de llegada.

Tabla 4.5: Resultados segunda prueba: Reglas de negocio

Tiempo Resolución (s)	1834
Gap Nodo Raíz	694.15 %
Función Objetivo	272.0719
Número Nodos	6735
Nodos Internos	3370
Nodos Singulares	3365
Profundidad Máxima	115
Número Rutas Creadas	225

4.2.3. Branching en el número de rutas por periodo

Siguiendo con la idea de mejorar el tiempo de resolución, tomamos la idea utilizada en Desaulniers et al. (2016) de hacer branching en el número de rutas usadas por periodo. Para hacer lo anterior se introducen en el problema maestro un nuevo tipo de variable y una nueva restricción. En detalle, al problema maestro se incorporan:

- Z_t, variable entera que indica la cantidad de rutas utilizadas en el periodo t.
- Las restricciones que unen la nueva variable Z_t con las variables de ruta x_{rt}:

$$\sum_{r \in R_{it}} x_{rt} = Z_t \qquad \forall t \in T \tag{4.2}$$

La restricción (4.2), al incluir variables de ruta, impacta el subproblema de generación de columnas, el cual ahora debe ser modificado para incluir la variable dual irrestricta asociada a cada una de estas nuevas restricciones (una por periodo). Sin embargo, su inclusión es trivial, puesto que como dicha dual no esta asociada a un cajero o ruta particular sino al total de rutas en un periodo determinado, se puede ingresar en la función objetivo del subproblema de *pricing* como una constante, sin alterar el resto del problema.

La tabla 4.6 muestra los resultados de la implementación de lo anteriormente descrito, en conjunto con las restricciones de regla de negocio discutidas en el apartado anterior. Se observa una disminución de un 60 % en el tiempo de resolución a 12 minutos y 12 segundos (732 segundos), el tamaño del árbol de ramificación también se ha visto reducido, retrocediendo hasta un total de 1672 nodos (caída de un 75.2 %). Sin embargo, el gap en el nodo raíz ha vuelto a crecer, retrocediendo a niveles similares a los de la prueba inicial.

Tabla 4.6: Resultados tercera prueba: Branching en número de rutas por periodo

Tiempo Resolución (s)	732
Gap Nodo Raíz	830.1 %
Función Objetivo	272.0719
Número Nodos	1672
Nodos Internos	839
Nodos Singulares	833
Profundidad Máxima	92
Número Rutas Creadas	225

Combinando ambas mejoras el tiempo de resolución para la instancia de 3 cajeros y 3 periodos pasó de 56 minutos y 36 segundos a 12 minutos y 12 segundos representando una caída del 78.4 %, el tamaño del árbol de ramificación paso de tener 8941 nodos a 1672 nodos, una diferencia de un 81.3 %.

Luego de ambas mejoras, evaluamos el comportamiento del algoritmo en instancias un poco más grandes. Se incrementa el número de periodos inicialmente, como tope se logra resolver hasta 6 periodos. La tabla de demandas para dichos cajeros en tales periodos se encuentra en la tabla 4.7.

Tabla 4.7: Inventario inicial y demandas hasta 6 periodos

Cajero	Inventario Inicial	Periodo					
		T1	T2	T3	T4	T5	T6
C1	5.0	4.01	3.34	0.97	2.34	3.68	0.82
C2	5.0	7.93	5.28	1.53	5.28	4.76	0.94
C3	1.0	1.62	1.22	0.30	0.68	1.62	0.39

Dada la baja demanda en los cajeros en los periodos 4 a 6 en los 3 cajeros, la solución en todas las instancias es la misma que en la instancia de la prueba inicial. La misma ruta visita los 3 cajeros en el primer periodo, saliendo al inicio del periodo, entregando el *cassette* pequeño en las tres visitas (mismos valores que en la tabla 4.4). La tabla 4.8 muestra la solución de la instancia de 3 cajeros y 6 periodos, esta muestra los inventarios de los 3 cajeros a través del horizonte de tiempo, se ve que sólo hay una visita a cada cajero y luego los inventarios se reducen periodo a periodo según su demanda.

En la tabla 4.9 podemos encontrar un resumen de los resultados de dichas instancias. Lo primero que se puede notar es el rápido crecimiento de los tiempos de resolución, multiplicándose varias veces a medida de que aumenta el número de periodos, de hecho la instancia con 6 periodos sube a un total de 3 horas y 46 minutos. Lamentablemente, al revisar la cantidad de rutas totales al final de cada instancia, se observa que el esquema genera todas las rutas factibles para las instancias resueltas.

Tabla 4.8: Inventarios en la solución a instancia de 3 cajeros y 6 periodos

Inv. Inicial	Cajero	Periodo	Inv. Final	Demanda	Cassette	T. Llegada
5.00	C1	T1	36.29	4.01	40.0	2178
36.29	C1	T2	32.95	3.34	-	-
32.95	C1	T3	31.98	0.97	-	-
31.98	C1	T4	29.64	2.34	-	-
29.64	C1	T5	25.96	3.68	-	-
25.96	C1	T6	25.14	0.82	-	-
5.00	C2	T1	32.23	7.93	40.0	576
32.23	C2	T2	26.95	5.28	-	-
26.95	C2	T3	25.42	1.53	-	-
25.42	C2	T4	20.14	5.28	-	-
20.14	C2	T5	15.38	4.76	-	-
15.38	C2	T6	14.44	0.94	-	-
1.00	C3	T1	38.55	1.62	40.0	3060
38.55	C3	T2	37.33	1.22	-	-
37.33	C3	T3	37.03	0.30	-	-
37.03	C3	T4	36.35	0.68	-	-
36.35	C3	T5	34.73	1.62	-	-
34.73	C3	T6	34.34	0.39	-	-

Al igual que los tiempos de resolución, el tamaño de los árboles de ramificación crece a un ritmo acelerado, con un tamaño de casi 16 mil nodos en la instancia con 6 periodos. Esto se conecta con la cantidad de rutas (variables sobre las cuales la regla de ramificación puede ramificar) que terminan en el problema maestro, que al ver el set de rutas que genera el algoritmo al ser el problema resuelto, estas representan el total de rutas factibles para la instancia. Este no es un resultado deseable y la primera duda es si la totalidad de las rutas se generan en el nodo raíz. En ninguna de las instancias se crearon más de 9 rutas en el nodo raíz y la mayor profundidad en que se creó una variable es de 9 nodos.

Tabla 4.9: Resultados instancias hasta 6 periodos

Métrica	Instancia			
	3C - 3P	3C - 4P	3C - 5P	3C - 6P
Tiempo Resolución (s)	732	2338	8408	13604
Gap Nodo Raíz	830.1 %	672.8 %	761.8 %	772.4 %
Función Objetivo	272.0719	315.1389	353.1758	390.1378
Número Nodos	1672	4295	13426	15969
Nodos Internos	839	2199	6721	7985
Nodos Singulares	833	2096	6705	7984
Profundidad Máxima	92	88	151	112
Número Rutas Creadas	225	300	375	450

Para investigar el problema de la excesiva generación de variables (rutas), se observa el

comportamiento de las duales y los componentes de la función objetivo del subproblema de generación de columnas en la expresión (3.64), en especial las duales asociadas a las restricciones del problema maestro que poseen las constantes M_i. Específicamente se pone atención a la sección de dicha expresión que acompaña a la variable x_{ip}:

$$M_i\left(\alpha_{it}+\beta_{it}-\phi_{it}\right)+L_p\left(\phi_{it}-\zeta_{it}-\gamma_{it}\right)-\mu_{it} \tag{4.3}$$

Dentro de la expresión (4.3) podemos ver 3 componentes: la asociada a la constante M_i, la asociada a la constante L_p y la dual μ_{it}. Las duales α_{it}, β_{it} y ϕ_{it}, al ver las restricciones (3.2), (3.3) y (3.5), sólo pueden tener un valor distinto de cero cuando no hay visitas al cajero en dicho periodo ni se incurre en un quiebre de stock (cuando están activas), lo cuál dadas las demandas es algo usual y, además, deseable desde un punto de vista operativo. Sin embargo al estar asociadas a la constante M_i que tiene un tamaño igual a la del *cassette* más grande (65 unidades pecuniarias) son sobredimensionadas en el subproblema de generación de columnas, facilitando la excesiva generación de rutas. De hecho, se calcula el comportamiento de la expresión (4.3) y se compara contra el valor de la componente de dicha expresión que esta asociada a la constante M_i, en adelante llamada *Big M*. En la tabla 4.10 vemos la comparación entre estos dos valores cuando las restricciones antes mencionadas están activas y cuando no lo están. Se puede apreciar que cuando están inactivas el valor de *Big M*, se queda en cero, tal como cabe de esperar, y el valor de x_{ip} es mayor a cero, indicando que no es recomendable visitar dicho cajero. Por otro lado, cuando las restricciones están activas el valor de *Big M* se torna negativo, reforzado por el alto valor de M_i y el valor de x_{ip} es, en la mayoría de los casos, igual al de *Big M*, tal como es apreciable en la tabla, haciendo que sea atractivo agregar dicho cajero en la ruta. En ocasiones, con el fin de encontrar una ruta factible (y que no esté ya creada) con costo reducido negativo, el subproblema agrega cajeros con valor x_{ip} positivos apalancándose en un cajero con valor x_{ip} muy negativo.

Tabla 4.10: Comparación entre componentes de la función objetivo del subproblema de generación de columnas

Restricciones (3.2), (3.3) y (3.5) están inactivas						
	Mín.	1er Qu.	Mediana	Media	3er Qu.	Máx.
x_{ip}	0.00	20.00	20.00	28.46	40.00	60.00
Big M	0.00	0.00	0.00	0.00	0.00	0.00

Restricciones (3.2), (3.3) y (3.5) están activas						
	Mín.	1er Qu.	Mediana	Media	3er Qu.	Máx.
x_{ip}	-3123.54	-35.66	-20.00	-62.92	-16.58	288.40
Big M	-3123.54	-35.69	-20.00	-68.16	-19.19	228.40

Lo descrito en el párrafo anterior se puede observar en la figura 4.4, donde encontrams la distribución de las diferencias entre x_{ip} y *Big M* cuando las restricciones anteriormente mencionadas están activas. Vemos que para los 3 cajeros se aglomeran en su gran mayoría cerca del cero o en el cero, la media esta representada en cada cajero como un punto amarillo. El efecto es más pronunciado en el primer cajero puesto que es el que menor diferencia

presenta entre su inventario inicial y sus demandas, es decir, puede sobrevivir más tiempo sin visita y sin caer en quiebre de stock que los otros dos.

Este problema hace que se generen más rutas de las necesarias (o deseadas) e impacta el proceso de ramificación ya que hay que evaluar más variables, incrementando el número de nodos, lo que finalmente puede traducirse en tiempos de resolución más largos.

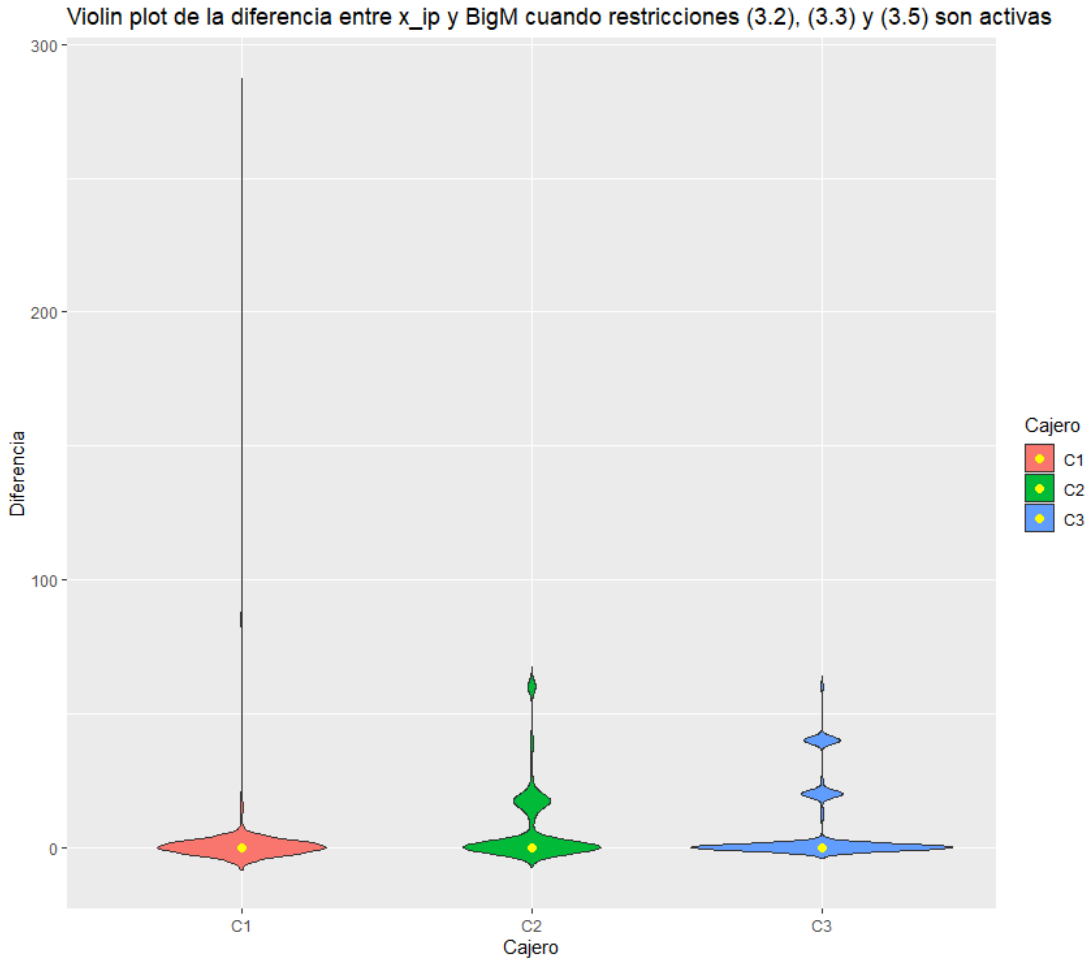


Figura 4.4: Gráfico de violín por cajero sobre la diferencia entre x_{ip} y $Big\ M$ cuando restricciones (3.2), (3.3) y (3.5) son activas

Este hallazgo y los gaps iniciales observados en los resultados ya mostrados indican que sería beneficioso una reformulación del problema maestro, y en ese intento, erradicar el uso de las constantes M_i del problema, lo cual no es tarea fácil dada la dinámica que se desea modelar en los inventarios, el comportamiento que siguen al haber visitas y quiebres en los cajero, encontrada en las expresiones (2.4) y (2.5).

4.3. Dinámicas de inventario y uso de salidas con retraso

En las pruebas anteriores dos aspectos particulares del modelo no se pudieron observar, los quiebres de inventario y la facilidad de iniciar rutas con retraso, aunque tampoco se logra apreciar con detalle el patrón de serrucho en la dinámica de inventarios puesto que en la prueba inicial todas las demandas podían ser satisfechas con una sola visita a cada cajero, la cual se realizaba en una sola ruta. Para poder ver los comportamientos antes mencionados, se modifican las demandas en los cajeros para forzar al algoritmo a mostrarnos su uso.

Se generan dos nuevos sets de demandas que incrementan los requerimientos de los clientes en periodos específicos. Este aumento se hace sin traspasar el valor del *cassette* más pequeño en ninguno de los periodos de modo de no transgredir los supuestos del problema. El valor escogido para los aumentos es de 35 unidades pecuniarias, y en periodos consecutivos, de modo forzar una segunda visita a los cajeros en donde la demanda ha sido aumentada. Las tablas 4.11 y 4.12 muestran las demandas en los dos nuevos escenarios y en ellas los cambios se encuentran resaltados en color azul.

Tabla 4.11: Demandas en segundo escenario

Cajero	Inv. Inicial	Periodo					
		T1	T2	T3	T4	T5	T6
C1	5.0	4.01	3.34	0.97	2.34	3.68	0.82
C2	5.0	7.93	5.28	1.53	5.28	4.76	0.94
C3	1.0	1.62	35.00	35.00	0.68	1.62	0.39

Tabla 4.12: Demandas en tercer escenario

Cajero	Inv. Inicial	Periodo					
		T1	T2	T3	T4	T5	T6
C1	5.0	4.01	3.34	0.97	35.00	35.00	0.82
C2	5.0	7.93	5.28	1.53	5.28	4.76	0.94
C3	1.0	1.62	35.00	35.00	0.68	1.62	0.39

4.3.1. Resultados de instancias del segundo escenario de demanda

Tal como en la prueba inicial, el algoritmo se corre tomando instancias que contemplan 3 a 6 periodos, para 3 cajeros. La tabla 4.13 contiene los resultados resumidos de las 4 instancias corridas con el segundo escenario de demanda. Si bien los gap en el nodo raíz disminuyeron el tiempo de resolución en las primeras 2 instancias aumentó, debido a que también los árboles de ramificación son más grandes (mayor cantidad de nodos que explorar y resolver). Lo anterior cambia en las instancias de 5 y 6 cajeros, en donde los tiempos de resolución son menores y también lo son la cantidad de nodos totales respecto a sus pares del escenario de demanda inicial (que podemos ver en la tabla 4.9). Se observa que el total de

rutas creadas en las instancias de este escenario de demanda son idénticos a los del escenario inicial, sugiriendo que el problema antes mencionado sigue ocurriendo.

Tabla 4.13: Resultados instancias del segundo escenario de demanda

Métrica	Instancia			
	3C - 3P	3C - 4P	3C - 5P	3C - 6P
Tiempo Resolución (s)	1671	3537	6488	10977
Gap Nodo Raíz	271.73 %	302.72 %	302.46 %	474.01 %
Función Objetivo	269.81	297.14	319.44	340.66
Número Nodos	3302	5194	7167	11266
Nodos Internos	1650	2604	3585	5641
Nodos Singulares	1652	2590	3582	5625
Profundidad Máxima	109	120	97	96
Número Rutas Creadas	225	300	375	450

Los óptimos sin embargo muestran algo que, a primera vista, parece contra intuitivo. Si bien hay una mayor demanda por efectivo, el costo total es inferior en todas las instancias resueltas. Esto se debe a que la mayor demanda en el tercer cajero disminuye el valor de los inventarios, reduciendo los costos de mantener tales saldos en los cajeros. Esto a pesar de que en este escenario es necesario ocupar dos rutas para satisfacer la demanda del cajero tres. La tabla 4.14 muestra el resultado de los inventarios de la instancia de tres cajeros y tres periodos, en el tercer cajero se muestra que este recibe dos entregas del *cassette* pequeño y que los valores de su inventario respecto a los vistos en la tabla 4.4 son notablemente inferiores, lo que genera la diferencia en la función objetivo respecto del escenario inicial.

Tabla 4.14: Inventarios solución instancia de 3C - 3P para segundo escenario de demanda

Inv. Inicial	Cajero	Periodo	Inv. Final	Demanda	Cassette	T. Llegada
5.00	C1	T1	36.29	4.01	40.0	2178
36.29	C1	T2	32.95	3.34	-	-
32.95	C1	T3	31.98	0.97	-	-
5.00	C2	T1	32.23	7.93	40.0	576
32.23	C2	T2	26.95	5.28	-	-
26.95	C2	T3	25.42	1.53	-	-
1.00	C3	T1	38.55	1.62	40.0	3060
38.55	C3	T2	3.55	35.00	-	-
3.55	C3	T3	5.55	35.00	40.0	456

La figura 4.5 nos muestra el patrón de serrucho que se deseaba modelar en este problema. Ambas visitas reemplazan el saldo en el inventario del cajero por el valor del *cassette* entregado en la visita, en este caso el *cassette* pequeño que entrega 40 unidades pecuniarias, para luego ver como el saldo es consumido a una tasa dependiente de la magnitud de la demanda del periodo observado (periodos con mayor demanda poseen una pendiente de consumo más negativa que periodos de menor demanda).

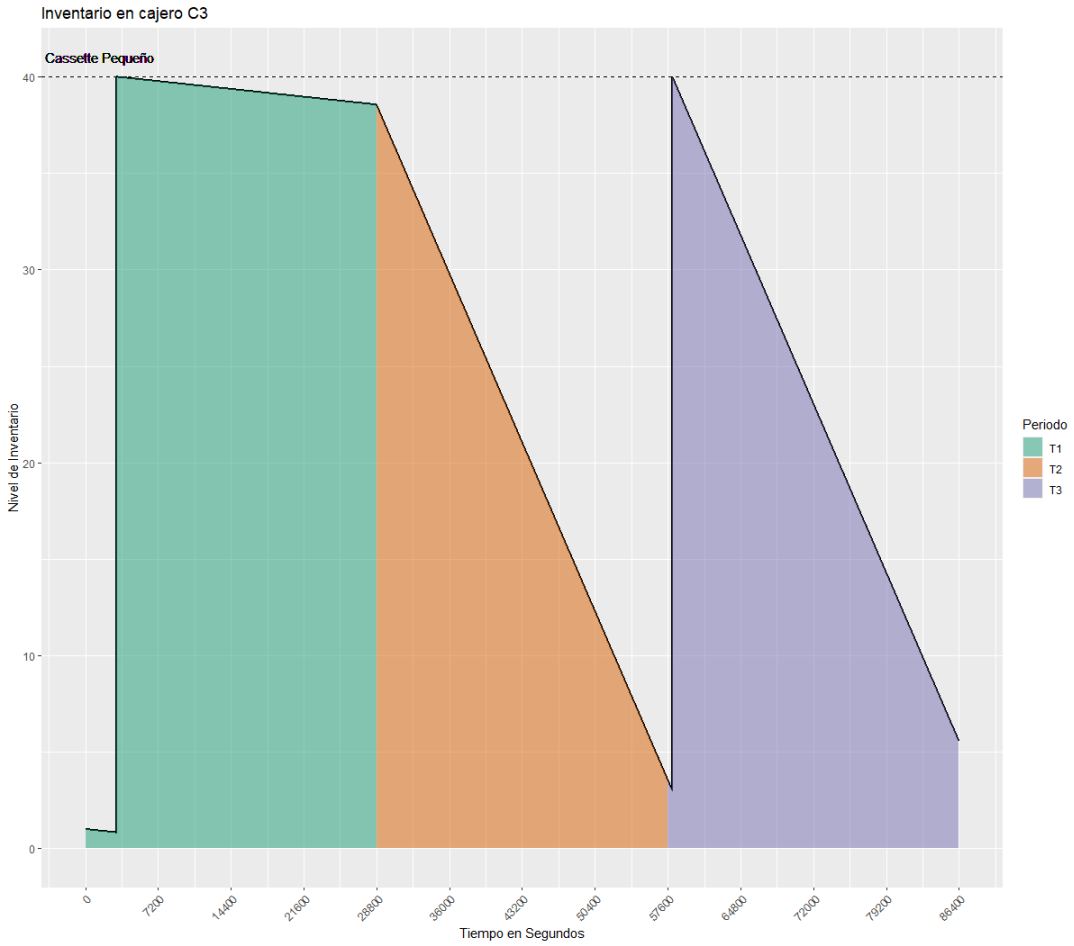


Figura 4.5: Evolución del inventario en el cajero 3 en segundo escenario de demanda

Si bien vemos comportamientos deseados, este escenario de demanda no nos provee de quiebres de inventario ni se observa el uso de la facilidad de salida con retraso. La tabla 4.15 muestra que justamente las dos rutas que se usan en la solución de la instancia de 3 periodos son suficientes para satisfacer la demanda de los periodos siguientes sin incurrir en quiebres de inventarios. El saldo en los cajeros disminuye según la demanda en cada periodo hasta el final del horizonte de tiempo.

Las figuras 4.6 y 4.7 nos muestran los recorridos de las rutas utilizadas en la solución de las instancias de este escenario de demanda. El depot en estas figuras se muestra en color rojo.

Tabla 4.15: Inventarios en la solución a instancia de 3C - 6P para segundo escenario de demanda

Inv. Inicial	Cajero	Periodo	Inv. Final	Demanda	Cassette	T. Llegada
5.00	C1	T1	36.29	4.01	40.0	2178
36.29	C1	T2	32.95	3.34	-	-
32.95	C1	T3	31.98	0.97	-	-
31.98	C1	T4	29.64	2.34	-	-
29.64	C1	T5	25.96	3.68	-	-
25.96	C1	T6	25.14	0.82	-	-
5.00	C2	T1	32.23	7.93	40.0	576
32.23	C2	T2	26.95	5.28	-	-
26.95	C2	T3	25.42	1.53	-	-
25.42	C2	T4	20.14	5.28	-	-
20.14	C2	T5	15.38	4.76	-	-
15.38	C2	T6	14.44	0.94	-	-
1.00	C3	T1	38.55	1.62	40.0	3060
38.55	C3	T2	3.55	35.00	-	-
3.55	C3	T3	5.55	35.00	40.0	456
5.55	C3	T4	4.87	0.68	-	-
4.87	C3	T5	3.25	1.62	-	-
3.25	C3	T6	2.86	0.39	-	-

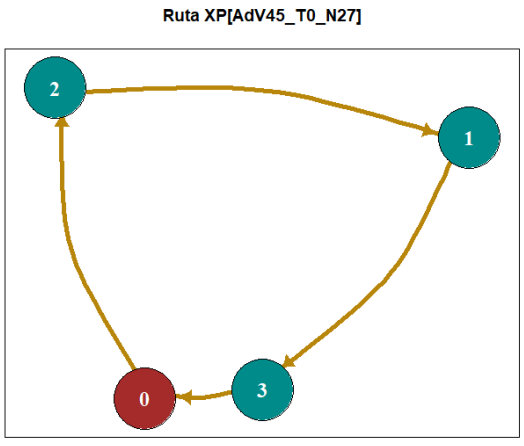


Figura 4.6: Primera ruta en solución del segundo escenario

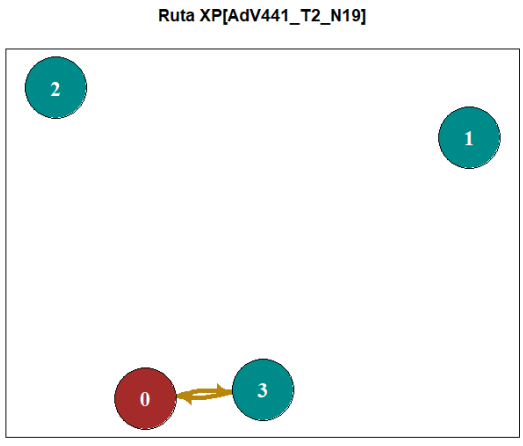


Figura 4.7: Segunda ruta en solución del segundo escenario

4.3.2. Resultados de instancias del tercer escenario de demanda

Los resultados de las cuatro instancias resueltas con el tercer escenario de demanda se resumen en la tabla 4.16. Se puede ver que los gap en el nodo raíz se redujeron respecto a los escenarios anteriores, pero los tiempos de resolución sólo disminuyen en las dos primeras instancias, lo mismo para el tamaño de los árboles. Lo primero que vemos al observar las soluciones es que en el caso de la instancia de 3 periodos la solución es la misma que en la encontrada en el segundo escenario de demanda, resultado esperable dado que para tal instancia las demandas son las mismas en ambos escenarios. Los cambios aparecen desde la instancia de 4 periodos en adelante.

Tabla 4.16: Resultados instancias del tercer escenario de demanda

Métrica	Instancia			
	3C - 3P	3C - 4P	3C - 5P	3C - 6P
Tiempo Resolución (s)	1467	1726	8835	16072
Gap Nodo Raíz	271.73 %	156.29 %	140.42 %	147.64 %
Función Objetivo	269.81	286.30	381.48	400.65
Número Nodos	3302	2636	9757	15869
Nodos Internos	1650	1316	4929	8030
Nodos Singulares	1652	1320	4828	7839
Profundidad Máxima	109	89	64	84
Número Rutas Creadas	225	300	375	450

En la instancia de 4 periodos la conformación de la solución cambia, dos rutas que realizan dos visitas cada una (se pueden ver en las figuras 4.8 y 4.9), por primera vez se entrega uno de los *cassette* de mayor valor (65 unidades pecuniarias) y se utiliza la facilidad que poseen las rutas de salir con atraso. En la ruta que visita los cajeros C1 y C3 en el segundo periodo. La ruta salió 41.3 minutos (2478 segundos) iniciado el segundo periodo.

Tabla 4.17: Inventarios en la solución a instancia de 3C - 4P para tercer escenario de demanda

Inv. Inicial	Cajero	Periodo	Inv. Final	Demanda	Cassette	T. Llegada
5.00	C1	T1	0.99	4.01	-	-
0.99	C1	T2	37.10	3.34	40.0	3791
37.10	C1	T3	36.13	0.97	-	-
36.13	C1	T4	1.13	35.00	-	-
5.00	C2	T1	32.23	7.93	40.0	576
32.23	C2	T2	26.95	5.28	-	-
26.95	C2	T3	25.42	1.53	-	-
25.42	C2	T4	20.14	5.28	-	-
1.00	C3	T1	38.46	1.62	40.0	1494
38.46	C3	T2	35.68	35.00	65.0	4674
35.68	C3	T3	0.68	35.00	-	-
0.68	C3	T4	0.00	0.68	-	-

La tabla 4.17 resume la solución de la instancia de 4 periodo, mostrando los momento de vista a cada cajero. Lo interesante que se puede observar es el inventario final del cajero C3 en el cuarto periodo, viendo que el algoritmo decide desplazar el inicio de la segunda ruta para hacer que el inventario del cajero termine en cero al final del horizonte de tiempo, minimizando los costos de inventario, tal como se puede apreciar en la figura 4.10.

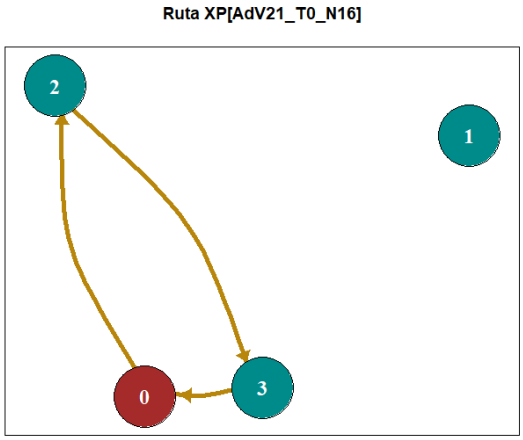


Figura 4.8: Primera ruta en solución de instancia 3C-4P del tercer escenario

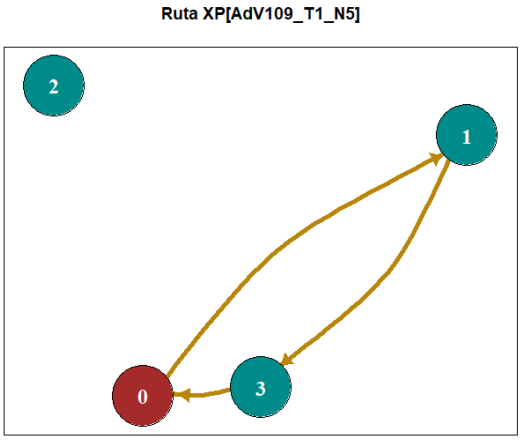


Figura 4.9: Segunda ruta en solución de instancia 3C-4P del tercer escenario

La solución de la instancia de cinco periodo utiliza rutas similares a las utilizadas en la solución de la instancia de 4 periodos. Dos rutas que visitan los mismo dos cajeros en los que sus símiles en la instancia anterior, la diferencia es que la segunda en vez de entregar un *cassette* de cada tipo, entrega dos *cassette* de tamaño grande a los dos cajeros que visita, tal como se puede apreciar en la tabla 4.18. La segunda ruta, tal como en la instancia de 4 periodos, sale con atraso, pero en esta ocasión el atraso con que dicha ruta inicia su recorrido es de 92 minutos, versus los 41 minutos de atraso de la ruta en la instancia anterior. Este mayor atraso lo hace para cubrir la demanda del periodo adicional para el que planifica el algoritmo en esta instancia.

En esta instancia, y por primera vez, se tiene que un quiebre de stock, este se da en el primer cajero en el último periodo. Este cajero cae en quiebre de stock a las 6 horas y 3 minutos de iniciado el quinto periodo, dejando insatisfecha un total de 8.51 unidades pecuniarias. este quiebre se puede observar al final de la figura 4.11. La razón por lo que se dejo que este cajero cayera en quiebre de stock es debido a que la suma del valor de una segunda visita y el de mantener su inventario era superior al costo incurrido de dejarlo quebrar y asumir el costo de la demanda no satisfecha.

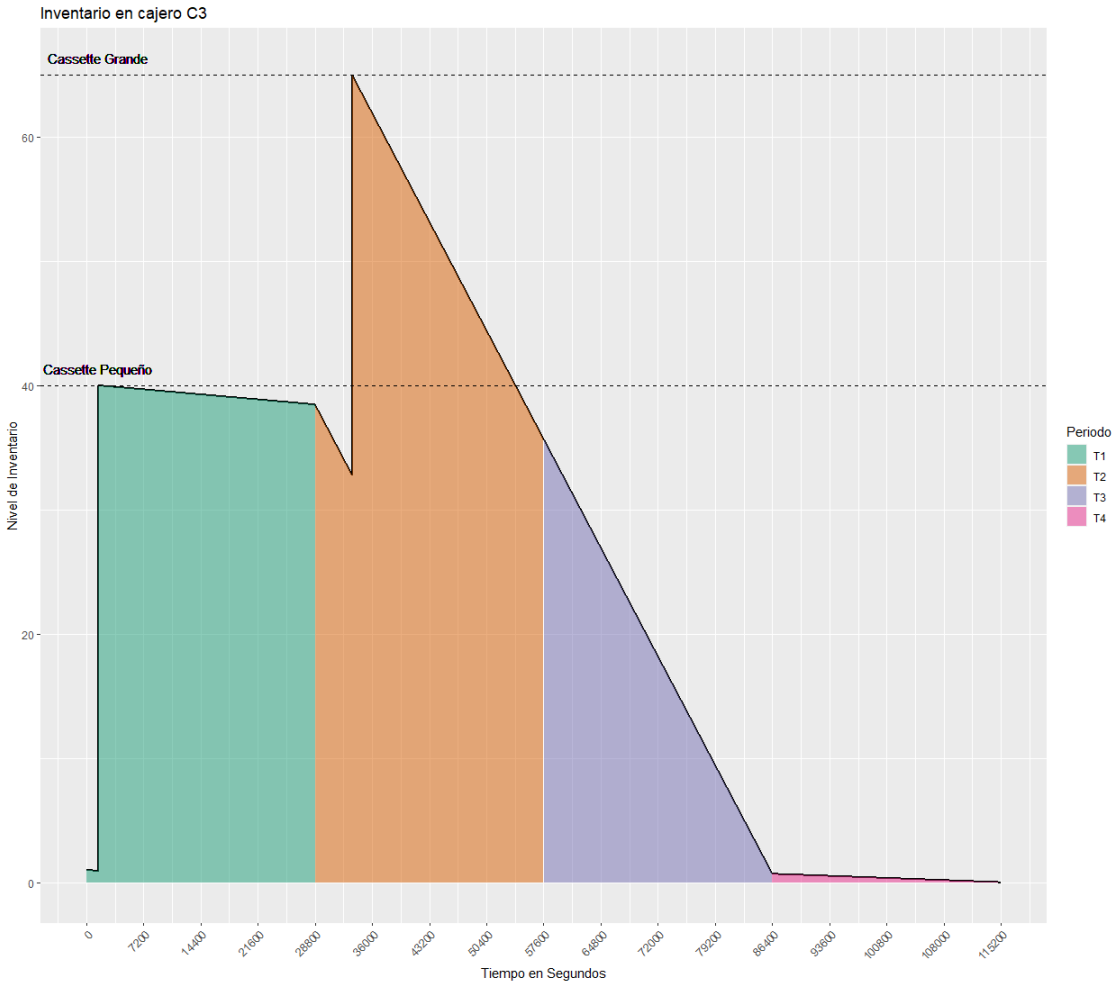


Figura 4.10: Evolución del inventario en cajero C3 en instancia 3C-4P del tercer escenario

En la última instancia, la de 6 periodos como horizonte de planificación, se realiza una tercera ruta, que visita al cajero que en la instancia anterior quiebra, ahora que al agregar un nuevo periodo (con elevada demanda) el costo de la demanda no satisfecha y quiebre supera el de realizar la visita y mantener el inventario. La figura 4.12 muestra cómo quedaría la evolución del inventario del primer cajero ahora que no cae en quiebre de stock, mientras que la tabla 4.19 nos muestra el momento de llegada de dicha ruta al cajero C1.

En esta instancia la ruta que visita los cajeros C1 y C3 en el segundo periodo vuelve a salir con retraso, pero esta vez el atraso aumenta 15 minutos y 50 segundos, para quedar en un total de 107 minutos y 50 segundos de atraso total. Nuevamente, este incremento en el atraso se debe a que con la misma visita desea extender la duración del inventario para considerar la demanda del nuevo periodo introducido al horizonte de planificación.

Tabla 4.18: Inventarios en la solución a instancia de 3C - 5P para tercer escenario de demanda

Inv. Inicial	Cajero	Periodo	Inv. Final	Demanda	Cassette	T. Llegada
5.00	C1	T1	0.99	4.01	-	-
0.99	C1	T2	62.46	3.34	65.0	6889
62.46	C1	T3	61.49	0.97	-	-
61.49	C1	T4	26.49	35.00	-	-
26.49	C1	T5	0.00	35.00	-	-
5.00	C2	T1	32.23	7.93	40.0	576
32.23	C2	T2	26.95	5.28	-	-
26.95	C2	T3	25.42	1.53	-	-
25.42	C2	T4	20.14	5.28	-	-
20.14	C2	T5	15.38	4.76	-	-
1.00	C3	T1	38.46	1.62	40.0	1494
38.46	C3	T2	37.30	35.00	65.0	6007
37.30	C3	T3	2.30	35.00	-	-
2.30	C3	T4	1.62	0.68	-	-
1.62	C3	T5	0.00	1.62	-	-

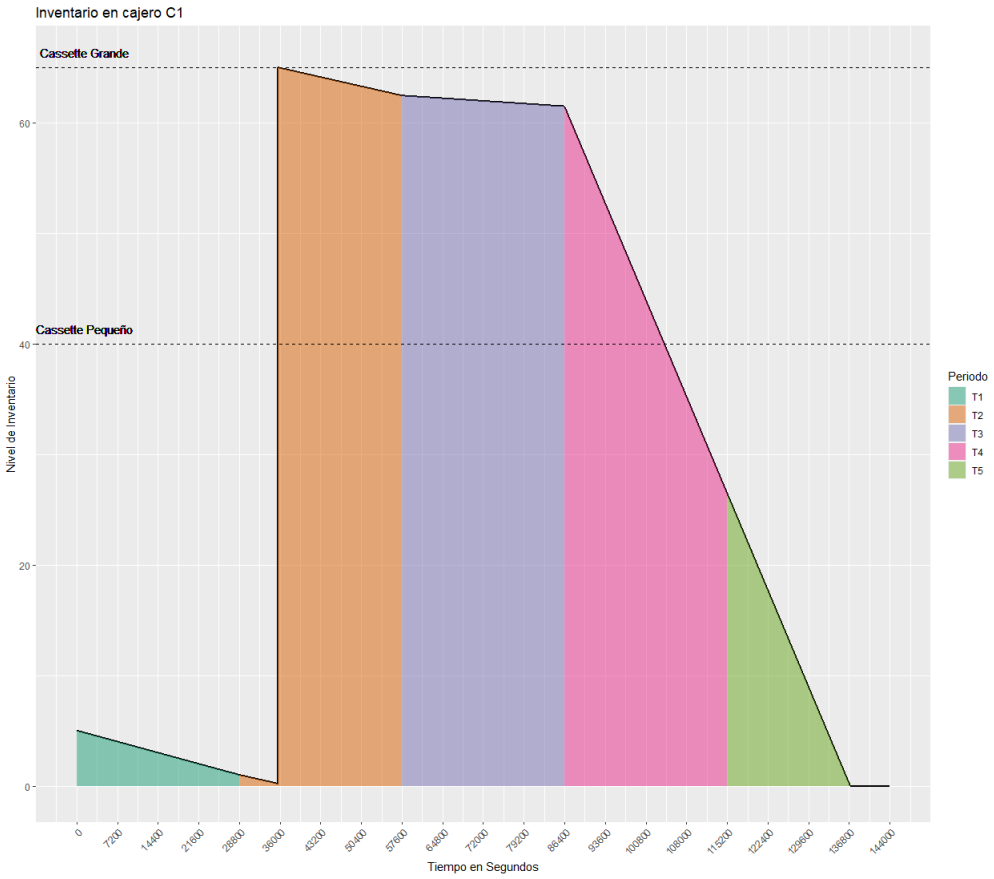


Figura 4.11: Evolución del inventario en cajero C1 en instancia 3C-5P del tercer escenario

Tabla 4.19: Inventarios en la solución a instancia de 3C - 6P para tercer escenario de demanda

Inv. Inicial	Cajero	Periodo	Inv. Final	Demanda	Cassette	T. Llegada
5.00	C1	T1	0.99	4.01	-	-
0.99	C1	T2	37.57	3.34	40.0	7809
37.57	C1	T3	36.60	0.97	-	-
36.60	C1	T4	1.60	35.00	-	-
1.60	C1	T5	6.60	35.00	40.0	1313
6.60	C1	T6	5.78	0.82	-	-
5.00	C2	T1	32.23	7.93	40.0	576
32.23	C2	T2	26.95	5.28	-	-
26.95	C2	T3	25.42	1.53	-	-
25.42	C2	T4	20.14	5.28	-	-
20.14	C2	T5	15.38	4.76	-	-
15.38	C2	T6	14.44	0.94	-	-
1.00	C3	T1	38.46	1.62	40.0	1494
38.46	C3	T2	38.42	35.00	65.0	6926
38.42	C3	T3	3.42	35.00	-	-
3.42	C3	T4	2.74	0.68	-	-
2.74	C3	T5	1.12	1.62	-	-
1.12	C3	T6	0.73	0.39	-	-

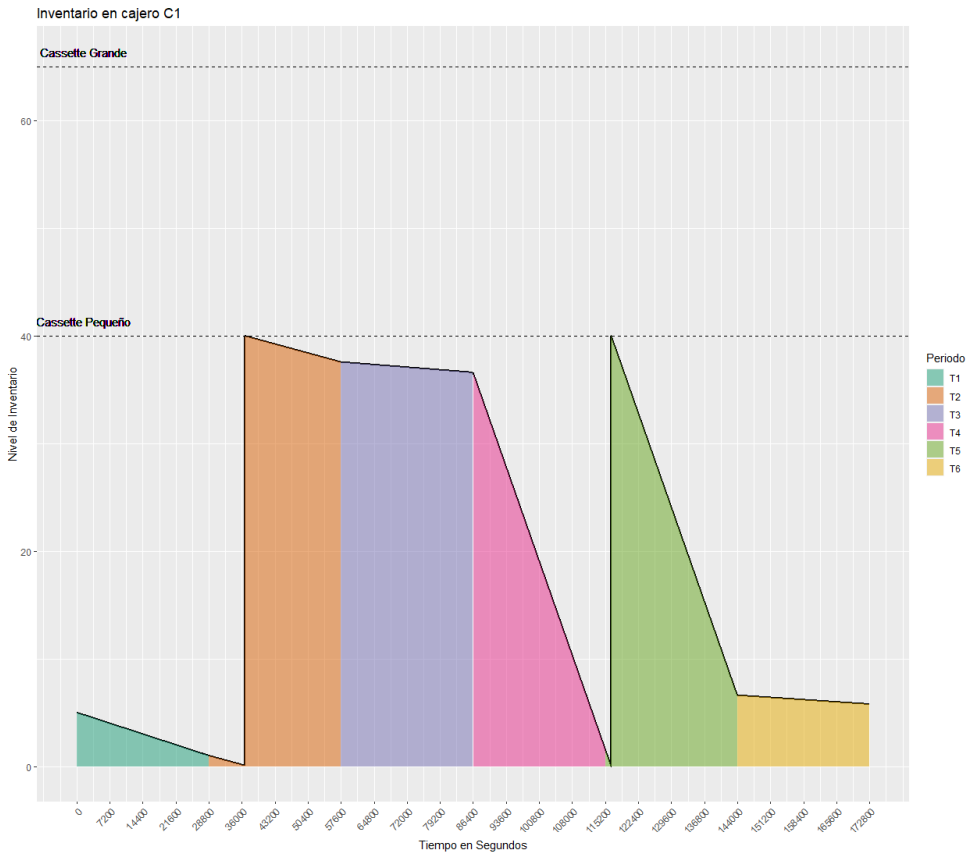


Figura 4.12: Evolución del inventario en cajero C1 en instancia 3C-6P del tercer escenario

Como se pudo observar de los resultados de las instancias corridas para los escenarios dos y tres, el modelo se comporta de forma esperada y reproduce la dinámica de inventario que se deseaba modelar, al punto de que es posible identificar los momentos exactos en que suceden los eventos durante el horizonte de planificación que se le proponga, al detalle de tener los segundos en que se visita cada cajero y el segundo en que se incurren en quiebre de stock.

Las decisiones tomadas por el algoritmo son correctas y acertadas, como pudimos ver en cuanto variamos los requerimientos de demanda, permitiendo quiebres cuando es conveniente y desplazando el inicio del recorrido de las rutas de forma de cubrir mejor las necesidades de los clientes de la red de cajeros de forma costo efectiva. Sin embargo se identificaron debilidades dentro de la modelación del problema, en donde el uso de las constantes M_i en las restricciones (3.2), (3.3) y (3.5) del problema maestro impactan negativamente el comportamiento del subproblema de generación de columnas, obteniendo de esta forma una cantidad indeseada de rutas que no son parte de la solución final de las instancias. Adicionalmente, a lo largo de las diversas instancias resueltas se observa que la relajación lineal del maestro es débil, como nos dejaron en claro los gap en el nodo raíz del problema, indicando que una modificación podría ser altamente beneficiosa.

4.4. Alteración del tamaño de *Cassette* y reglas sobre el nivel de servicio

Dentro de la siguiente variación del problema se realizan dos modificaciones y se agrega una restricción al problema maestro. Los cambios realizados afectan los tamaños de *cassette*, las restricciones sobre el nivel de servicio y el número mínimo de rutas que se deben realizar en la solución de cada instancia.

La primera modificación, sobre el tamaño de los *cassette*, se realiza con el fin de acercar sus tamaños lo más posible a los valores observados en las demandas en los cajeros. Mirando la tabla 4.1, la demanda máxima para un cajero en un periodo particular es de 9.06 unidades pecuniarias, ahora, y tomando en cuenta el supuesto del problema de que el tamaño del *cassette* pequeño debe poder cubrir la demanda de todo cajero en cualquier periodo, el tamaño escogido para el valor de *cassette* pequeño es de 10. Para el *cassette* de mayor tamaño aumentaremos el valor anterior en dos veces la media observada, lo que redondeando nos da un valor de 15 unidades pecuniarias. Adicionalmente, dado que se modificó el tamaño de los *cassette*, la carga máxima permitida por ruta se modifica a un total de 75 unidades pecuniarias.

La segunda modificación afecta las restricciones (3.11) a (3.13) del problema maestro. El nivel de servicio dejará de medirse según la cantidad de cajeros-periodos en que se incurre en quiebre de stock (variables binarias y_{it}) y pasará a medirse según la cantidad de demanda perdida (variable continua o_{it}). Las reglas son las mismas respecto al porcentaje máximo de incumplimiento de demanda: 50 % por cajero y/o periodo y 25 % en total. Las restricciones

de nivel de servicio finalmente quedan de la siguiente manera:

$$\sum_{i \in I'} o_{it} \leq 0{,}5 \sum_{i \in I'} \lambda_{it} \qquad \forall t \in T \tag{4.4}$$

$$\sum_{t \in T} o_{it} \leq 0{,}5 \sum_{t \in T} \lambda_{it} \qquad \forall i \in I' \tag{4.5}$$

$$\sum_{i \in I'} \sum_{t \in T} o_{it} \leq 0{,}25 \sum_{i \in I'} \sum_{t \in T} \lambda_{it} \tag{4.6}$$

Ahora, la restricción que se agrega pone una cota inferior al número de rutas que deben utilizarse con el fin de satisfacer el nivel de servicio total exigido en la instancia que se este resolviendo, encontrada en la restricción (4.6). Esto se realiza utilizando las nuevas variables Z_t utilizadas en la restricción (4.2). El número mínimo de rutas necesarias (N^{RM}) se obtiene utilizando la demanda total en la instancia, los inventarios iniciales en cada cajero (θ_i), la capacidad máxima permitida en cada ruta ($\overline{Q}$) y la fracción de demanda que debe ser atendida como mínimo (75 %):

$$N^{RM} = \frac{0{,}75 \left[\sum_{i \in I'} \sum_{t \in T} \lambda_{it} - \sum_{i \in I'} \theta_i\right]}{\overline{Q}} \tag{4.7}$$

De modo que la restricción sobre el mínimo de rutas que deben ser utilizadas en la solución de cada instancia queda de la siguiente forma:

$$\sum_{t \in T} Z_t \geq N^{RM} \tag{4.8}$$

Con estos cambios, se logra resolver 3 nuevas instancias y mejorar las estadísticas de las pruebas anteriores, tal como la tabla 4.20 muestra.

Tabla 4.20: Resultados de instancias con *cassette* más pequeños y nuevas restricciones de nivel de servicio

Métrica	Instancia						
	3C - 3P	3C - 4P	3C - 5P	3C - 6P	3C - 7P	3C - 8P	4C - 3P
Tiempo Resolución(s)	284	1341	2191	1760	6539	31588	36978
Gap Nodo Raíz	190.8 %	118.5 %	127.3 %	120.5 %	105.3 %	103.6 %	135.6 %
Función Objetivo	144.57	187.3	201.26	204.57	249.91	292.43	156.12
Número Nodos	563	2149	2371	1733	6152	28009	6099
Nodos Internos	282	1083	1273	867	3078	14390	3049
Nodos Singulares	281	1066	1098	866	3074	13619	3050
Profundidad Máxima	105	71	92	77	87	63	606
Núm. Rutas Creadas	225	300	375	450	525	600	1884

Comparando con la tabla del tercer escenario de demanda 4.16, la tabla con mejores resultados hasta entonces, vemos un gran reducción en el tamaño de los gaps iniciales y

en los tiempos de resolución. El tamaño de los árboles de ramificación también se redujo significativamente pero de forma más moderada que los tiempos de resolución. La mejora de esta versión del esquema permitió la resolución de dos nuevas instancias con tres cajeros y una con 4 cajeros. La última, la instancia de cuatro cajeros y 3 periodos, anteriormente había necesitado un total de 18 horas para ser resuelta versus las 10 que ahora toma, sin embargo esto siegue siendo un tiempo de resolución extremadamente largo. Por lo demás, nuevamente se observa que la cantidad de rutas generadas en cada instancia es grande y es exactamente igual a la cantidad de rutas factibles, tal como sucedía en pruebas anteriores.

La cantidad de rutas empleadas y la cantidad de veces en que se utiliza la facilidad de diferir la salida de los vehículos desde el depot aumenta respecto a la instancia inicial. Como se puede ver en la tabla 4.8 esta instancia sólo necesita una visita, al igual todas las instancias anteriores a esa. En las instancias de la tabla 4.20, desde la instancia *3C-5P* todas necesita al menos 2 visitas, mientras que la de 8 periodos necesita 4 rutas y todas empiezan su recorrido de forma diferida.

Conclusión

En esta tesis se ha logrado el desarrollo e implementación de un esquema de *Branch & Price* para el problema de ruteo e inventario que concierne una red de cajeros automáticos. Asume y resuelve las dificultades inherentes a un problema clásico de IRP e incorpora las particularidades específicas del problema, descritas en el capítulo 2. Dentro de dichas particularidades se encuentran la dinámica específica de inventario en cada cajero automático, en donde se reemplaza el inventario completo en cada visita (intercambiando *cassette* de efectivo), la facultad de los vehículos de iniciar sus rutas con atraso, es decir, en un instante distinto al del inicio del periodo en cuestión, lo cual forzó a indagar en la literatura de *column-dependent-rows*, y la facilidad de identificar el momento exacto de cada visita que se realiza.

Los resultados mostraron que si bien el esquema se comporta como uno esperaría en el sentido que es posible observar el comportamiento deseado en los inventarios, se utilizan las facilidades entregadas con el fin de minimizar costes y es posible determinar los instantes específicos de llegada de cada vehículo, existe espacio para mejorar de forma extensa el esquema desarrollado. Dentro de los problemas identificados en el capítulo de resultados está el hecho de que la relajación lineal del problema maestro es muy débil, cosa que queda en evidencia al observar las distancias entre las cotas primal y dual de las diversas instancias resueltas. También esta el problema asociado a las constantes M_i, ligadas a las restricciones de inventario, que cuando son activas (casos en que no hay visita ni quiebre en el cajero y periodo de la restricción que se hable) impactan de forma negativa el subproblema de generación de columnas, facilitando la excesiva creación de columnas con costo reducido negativo, llegando al extremo de generar todas las rutas factibles en cada instancia resuelta.

Dado lo aprendido en el capítulo de resultados, es posible entrever varias líneas de trabajo futuro, tomando en cuenta de que el esquema de *Branch & Price* ya se encuentra armado y funcionando. El primero, como es de esperar, es una modificación del problema maestro con el fin de erradicar las contantes M_i, de modo de atacar el problema encontrado en este capítulo y la debilidad de la relajación lineal del problema maestro.

En otra avenida de investigación, la implementación de una regla de ramificación que no sea basada en variables, pero sí en columnas. En ese aspecto reglas para problemas tipo *set partitioning*, como la de *Ryan & Foster* (Ryan and Foster, 1981) pueden resultar útiles, determinando pares de cajeros sobre los cuales ramificar, con lo que se podría reducir los tiempos de resolución y atacar instancias de mayor tamaño, puesto que el número de pares factibles de cajeros crece de manera más lenta que el número total de rutas factibles a medida

que se agregan cajeros a las instancias que se desea resolver.

Respecto del subproblema de generación de columnas, incluir métodos adicionales al actual MIP para encontrar rutas de costo reducido negativo mejoraría el rendimiento del esquema. Heurísticas de inserción, ahorros o de asignación primero y ruteo después, o métodos más sofisticados como algunas de las matheurísticas discutidas en Speranza and Archetti (2014), para generar rápidamente columnas podría reducir los tiempos de resolución. Estas técnicas se utilizarían en una primera instancia de forma de encontrar rutas rápidamente, y cuando éstas fallen y la búsqueda se haga más compleja, comprobar vía MIP si es que no quedan rutas de costo reducido negativo que pudiesen agregarse al problema maestro.

Ninguna de las líneas de investigación propuestas viene sin su dificultad, puesto que deben tener en cuenta en su implementación el efecto que tengan en el subproblema de generación de columnas y todas las particularidades del problema, sin embargo el impacto que puedan tener es significativo y valen la pena ser abordados.

Finalmente, y atendiendo a la aplicabilidad del método desarrollado, el modelo propuesto incorpora gran parte de las características de la operación real para este tipo de problemas, permitiendo quiebres de inventarios penalizados en cajeros automáticos, incorporando restricciones tanto en la cantidad de dinero que lleva cada ruta como el tiempo total que un camión puede estar fuera del depot por temas de seguridad y la forma en que los inventarios son repuestos, vía *cassettes* con un monto fijo de dinero que deben reemplazarse en su integridad, indistinto de si el *cassette* reemplazado posee efectivo remanente o no. Todo lo anterior es aprendido gracias al trabajo de Larrain et al. (2017), que trabajan con instancias facilitadas por un operador de la ciudad de Santiago, Chile. Adicionalmente, se indica que el operador agrupa geográficamente cajeros (grupos pequeños) y hace entregas de iguales cantidades de dinero cuando estos son visitados en conjunto, de forma de que estos lleven similares niveles de inventario, de modo que el método usado es heurístico y para un horizonte de tiempo limitado. Ahora, el método de esta tesis es de solución exacta y, como es visto en los resultados reportados, toma una cantidad de tiempo considerable a medida que las instancias abordan más cajeros y horizontes mas extensos de planificación, en parte debido a los problemas descritos en párrafos anteriores. En consecuencia sería difícil aplicar el método en la vida real sin antes abordar algunas de las líneas de investigación futuras antes propuestas o el desarrollo de un esquema mate-heurístico que pueda resolver instancias de mayor tamaño agrupando cajeros y sus inventarios, quizás incluso, de forma jerárquica.

Bibliografía

Achterberg, T., 2007. Constraint integer programming. Ph.D. thesis, Technischen Universität Berlin.

Achterberg, T., Jul 2009. Scip: solving constraint integer programs. Mathematical Programming Computation 1 (1), 1–41.
URL https://doi.org/10.1007/s12532-008-0001-1

Achterberg, T., Berthold, T., Koch, T., Wolter, K., 2008. Constraint integer programming: A new approach to integrate cp and mip. In: Perron, L., Trick, M. A. (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 6–20.

Achterberg, T., Koch, T., Martin, A., 01 2005. Branching rules revisited. Operations Research Letters 33, 42–54.

Andersson, H., Hoff, A., Christiansen, M., Hasle, G., Lokketangen, A., 2010. Industrial aspects and literature survey: Combined inventory management and routing. Computers and Operations Research 37 (9), 1515–1536.
URL http://www.sciencedirect.com/science/article/pii/S0305054809002962

Appelgren, L. H., 1969. A column generation algorithm for a ship scheduling problem. Transportation Science 3 (1), 53–68.
URL http://www.jstor.org/stable/25767508

Applegate, D., Bixby, R., Chvatal, V., Cook, B., 1995. Finding cuts in the tsp (a preliminary report). Tech. rep.

Archetti, C., Bertazzi, L., Hertz, A., Speranza, M., 02 2012. A hybrid heuristic for an inventory routing problem. INFORMS Journal on Computing 24.

Archetti, C., Bertazzi, L., Laporte, G., Speranza, M., 08 2007. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transportation Science 41, 382–391.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., H. Vance, P., 01 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46, 316–.

Batl, S., Gözüpek, D., Dec 2019. Joint optimization of cash management and routing for

new-generation automated teller machine networks. IEEE Transactions on Systems, Man, and Cybernetics: Systems 49 (12), 2724–2738.

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., Prutzman, P. J., 1983. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. Interfaces 13 (6), 4–23.
URL http://www.jstor.org/stable/25060491

Bénichou, M., Gauthier, J.-M., Girodet, P., Hentges, G., Ribière, G., Vincent, O., 1971. Experiments in mixed-integer linear programming. Mathematical Programming 1, 76–94.

Bertazzi, L., Coelho, L. C., Maio, A. D., LaganÃ, D., 2019. A matheuristic algorithm for the multi-depot inventory routing problem. Transportation Research Part E: Logistics and Transportation Review 122, 524 – 544.
URL http://www.sciencedirect.com/science/article/pii/S1366554518307749

Bertazzi, L., Speranza, M., 01 2012. Matheuristics for Inventory Routing Problems. pp. 1–14.

Bertsimas, D., Tsitsiklis, J., 01 1998. Introduction to Linear Optimization.

Campbell, A. M., Savelsbergh, M. W. P., 2004. A decomposition approach for the inventory-routing problem. Transportation Science 38 (4), 488–502.
URL https://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0054

Chotayakul, S., Charnsetthikul, P., Pichitlamken, J., Kobza, J., 2013. An optimization-based heuristic for a capacitated lot-sizing model in an automated teller machines network. Journal of Mathematics and Statistics 9 (4), 283.

Christiansen, M., 1999. Decomposition of a combined inventory and time constrained ship routing problem. Transportation Science 33 (1), 3–16.
URL http://www.jstor.org/stable/25768842

Coelho, L., Laporte, G., 11 2013a. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. International Journal of Production Research 51.

Coelho, L., Laporte, G., 02 2013b. The exact solution of several classes of inventory-routing problems. Computers & Operations Research 40, 558âĂŞ565.

Coelho, L. C., Cordeau, J.-F., Laporte, G., 2014. Thirty years of inventory routing. Transportation Science 48, 1–19.

Coelho, L. C., Laporte, G., 2014. Improved solutions for inventory-routing problems through valid inequalities and input ordering. International Journal of Production Economics 155, 391 – 397, celebrating a century of the economic order quantity model.
URL http://www.sciencedirect.com/science/article/pii/S0925527313005343

Cordeau, J.-F., Lagana, D., Musmanno, R., Vocaturo, F., 2015. A decomposition-based heuristic for the multiple-product inventory-routing problem. Computers & Operations Research 55, 153 – 166.

URL http://www.sciencedirect.com/science/article/pii/S0305054814001658

Dakin, R. J., 01 1965. A tree-search algorithm for mixed integer programming problems. The Computer Journal 8 (3), 250–255.
URL https://doi.org/10.1093/comjnl/8.3.250

Dantzig, G. B., Ramser, J. H., 1959. The truck dispatching problem. Management Science 6 (1), 80–91.
URL https://doi.org/10.1287/mnsc.6.1.80

Desaulniers, G., 02 2010. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. Operations Research 58, 179–192.

Desaulniers, G., Rakke, J., Coelho, L., 10 2016. A branch-price-and-cut algorithm for the inventory-routing problem. Transportation Science 50.

Desrosiers, J., Lübbecke, M. E., 2005. A Primer in Column Generation. Springer US, Boston, MA, pp. 1–32.
URL https://doi.org/10.1007/0-387-25486-2_1

Dror, M., Ball, M., 1987. Inventory/routing: Reduction from an annual to a short-period problem. Naval Research Logistics (NRL) 34 (6), 891–905.

Feillet, D., 12 2010. A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR 8, 407–424.

Feillet, D., Gendreau, M., Medaglia, A., Walteros, J., 09 2010. A note on branch-and-cut-and-price. Operations Research Letters 38, 346–353.

Forrest, J. J. H., Hirst, J. P. H., Tomlin, J. A., 1974. Practical solution of large mixed integer programming problems with umpire. Management Science 20 (5), 736–773.
URL http://www.jstor.org/stable/2630088

Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Le Bodic, P., Maher, S. J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D., Witzig, J., March 2020. The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin.
URL http://nbn-resolving.de/urn:nbn:de:0297-zib-78023

Gillett, B. E., Miller, L. R., 1974. A heuristic algorithm for the vehicle-dispatch problem. Operations Research 22 (2), 340–349.
URL https://doi.org/10.1287/opre.22.2.340

Golden, B., Assad, A., Dahl, R., 12 1984. Analysis of a large-scale vehicle routing problem with an inventory component. Large Scale Systems 7.

Grønhaug, R., Christiansen, M., 04 2009. Supply chain optimization for the liquefied natu-

ral gas business. Innovations in Distribution Logistics, Lecture Notes in Economics and Mathematical Systems 619, 195–218.

Grønhaug, R., Christiansen, M., Desaulniers, G., Desrosiers, J., 08 2010. A branch-and-price method for a liquefied natural gas inventory routing problem. Transportation Science 44, 400–415.

Johnson, E. L., 1989. Modelling and strong linear programs for mixed integer programming. Springer-Verlag, pp. 1–43.

Kao, G., Sewell, E., Jacobson, S., 04 2009. A branch, bound, and remember algorithm for the 1|ri|?ti scheduling problem. Journal of Scheduling 12, 163–175.

Koc, C., Erbas, M., Ozceylan, E., 01 2018. A rich vehicle routing problem arising in the replenishment of automated teller machines. International Journal of Optimization and Control: Theories and Applications 8, 276–287.

Laporte, G., Nobert, Y., Desrochers, M., 1985. Optimal routing under capacity and distance restrictions. Operations Research 33 (5), 1050–1073.
URL http://www.jstor.org/stable/170853

Larrain, H., Coelho, L. C., Cataldo, A., 2017. A variable mip neighborhood descent algorithm for managing inventory and distribution of cash in automated teller machines. Computers & Operations Research 85, 22 – 31.
URL http://www.sciencedirect.com/science/article/pii/S0305054817300758

Linderoth, J., Savelsbergh, M., 05 1999. A computational study of search strategies for mixed integer programming. INFORMS Journal on Computing 11, 173–187.

Little, J. D. C., Murty, K. G., Sweeney, D. W., Karel, C., 1963. An algorithm for the traveling salesman problem. Operations Research 11 (6), 972–989.
URL https://doi.org/10.1287/opre.11.6.972

Martin, A., 1999. Integer programs with block structure.

Mehrotra, A., Trick, M. A., 1996. A column generation approach for graph coloring. INFORMS Journal on Computing 8 (4), 344–354.
URL https://doi.org/10.1287/ijoc.8.4.344

Mitchell, J. E., 2009. Integer Programming: Branch and Cut Algorithms. Springer US, Boston, MA, pp. 1643–1650.
URL https://doi.org/10.1007/978-0-387-74759-0_287

Morrison, D. R., Jacobson, S. H., Sauppe, J. J., Sewell, E. C., 2016. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization 19, 79 – 102.
URL http://www.sciencedirect.com/science/article/pii/S1572528616000062

Muter, I., Birbil, S. I., Bülbül, K., 2013. Simultaneous column-and-row generation for large-

scale linear programs with column-dependent-rows. Mathematical Programming 142, 47–82.

Ryan, D., Foster, B., 01 1981. An integer programming approach to scheduling. Computer Scheduling of Public Transport 1, 269–.

Savelsbergh, M. W. P., 2009. Branch and price: Integer programming with column generation-Branch and Price: Integer Programming with Column Generation. Springer US, Boston, MA, pp. 328–332.
URL https://doi.org/10.1007/978-0-387-74759-0_58

Speranza, M., Archetti, C., 11 2014. A survey on matheuristics for routing problems. EURO Journal on Computational Optimization 2.

Van Anholt, R., Coelho, L., Laporte, G., F. A. Vis, I., 05 2015. An inventory-routing problem with pickups and deliveries arising in the replenishment of automated teller machines. Transportation Science Articles in Advance, 1–15.

Vance, P. H., Barnhart, C., Johnson, E. L., Nemhauser, G. L., May 1994. Solving binary cutting stock problems by column generation and branch-and-bound. Comput. Optim. Appl. 3 (2), 111?130.
URL https://doi.org/10.1007/BF01300970

Vanderbeck, F., Wolsey, L. A., 1996. An exact algorithm for ip column generation. Operations Research Letters 19 (4), 151 – 159.
URL http://www.sciencedirect.com/science/article/pii/0167637796000338

Zak, E. J., 2002. Row and column generation technique for a multistage cutting stock problem. Computers & Operations Research 29 (9), 1143 – 1156.
URL http://www.sciencedirect.com/science/article/pii/S0305054800001118