Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Ruteo de vehículos y asignación de conductores: un enfoque combinado

Tesis de Licenciatura

Damián Bakarcic y Gabriela Di Piazza

Directores de tesis: Dra. Isabel Méndez Díaz
Dra. Paula Zabala

Buenos Aires, septiembre de 2012

Resumen

El problema de planificar rutas para vehículos y sus correspondientes conductores se presenta en la práctica en áreas de logística como transporte de pasajeros y distribución de mercadería. En la actualidad, dada la complejidad y variedad de los sistemas de transporte y distribución, se hace imperativo el uso de herramientas de optimización para resolver esta clase de problemas.

En este trabajo se aborda un problema surgido de la práctica que consiste en planificar el retiro y entrega de un conjunto de pedidos de mercadería, cumpliendo con ciertas restricciones horarias, en función de un conjunto de vehículos disponibles. Asimismo, se debe planificar la asignación de los conductores a los vehículos que realizarán cada tarea. El problema conjuga características de tres problemas estudiados en la literatura como son: el *Vehicle Routing Problem*, el *Crew Scheduling Problem* y el *Pick-up and Delivery Problem with Time Windows*.

Para resolver este problema proponemos un modelo de programación lineal entera binaria que contempla en forma integrada la planificación de rutas de vehículos y la asignación de los correspondientes conductores. Debido a que el modelo resultante cuenta con una gran cantidad de variables, utilizamos la técnica de generación de columnas para generar los posibles recorridos a realizar por los vehículos.

El esquema de resolución propuesto particiona al problema en dos: el problema maestro y el subproblema de generación de columnas. Para el primero utilizamos un algoritmo *Branch and Cut*. Para el último desarrollamos un algoritmo basado en programación dinámica que busca el camino óptimo sobre una red con recursos y restricciones. Los caminos sobre dicha red representan las posibles rutas que pueden transitar los vehículos.

Además de la implementación original propuesta para el algoritmo de generación de columnas, desarrollamos optimizaciones sobre el mismo y realizamos pruebas para verificar los resultados de dichas optimizaciones. Finalmente, presentamos un conjunto de pruebas que evalúa el comportamiento del esquema de resolución general implementado.

Abstract

The problem of vehicle routing planning combined with crew scheduling arise in areas of logistics such as passenger transport and merchandise distribution. At present, given the complexity and variety of transport and distribution systems, it is imperative to use optimization tools to solve such problems.

This work addresses a real life problem involving the planning of the pick-up and delivery of a set of orders, according to certain time restrictions, based on a set of available vehicles. In addition, the allocation of crews to the vehicles which perform each task must also be scheduled. This problem combines features present in three well studied problems: the *Vehicle Routing Problem*, the *Crew Scheduling Problem* and the *Pick-up and Delivery Problem with Time Windows*.

To solve this problem we propose a binary integer linear programming model that provides integrated planning of vehicle routing and crew scheduling. Due to the large number of variables of the resulting model, we use column generation in order to generate the possible routes to be taken by the vehicles.

The proposed resolution framework gives rise to two problems: the master problem and the subproblem of column generation. To solve the first we use a *Branch and Cut* algorithm. For the latter we develop a dynamic programming algorithm that seeks the optimal path over a resource constrained network. The paths on this network represent the possible routes that can be taken by the vehicles.

In addition to the original implementation, we develop optimizations over the column generation algorithm and perform tests to verify the results of these optimizations. Finally, we present a set of tests to analyze the performance of the implemented algorithms.

Agradecimientos

Nos gustaría agradecer en primer lugar a Isabel y Paula, nuestras directoras, por la dedicación, la paciencia y el apoyo que nos brindaron a lo largo de todo el trabajo.

A todos nuestros compañeros de la carrera que terminaron convirtiéndose en grandes amigos: Ale, Alex, Cele, Flor, Jona, Juli, Mati, Maxi, Nati, Pablo, Tom y todos aquellos que nos acompañaron en esta etapa. Con ellos compartimos cursadas, incontables horas de estudio, trabajos prácticos y, por sobre todo, muy buenos momentos. Gracias a su apoyo, todos estos años de estudio se hicieron mucho más agradables.

A los compañeros del Ministerio de Economía, Bruno, Caro, Cele, Fer, Flor, Mati y Tom, que nos aguantaron durante casi toda la carrera y nos ayudaron a crecer profesionalmente.

A los profesores y ayudantes del Departamento de Computación, por estar siempre dispuestos a transmitirnos su conocimiento y responder nuestras dudas. A la Universidad de Buenos Aires, por habernos recibido y brindado educación de excelencia.

Agradecimientos de Damián

Quería agradecer en primer lugar a mis viejos, Darko y Neris, que desde chiquito me motivaron y me dieron la oportunidad de aspirar a una carrera profesional, bancándome durante todos estos años de carrera. Espero poder devolverles algún día todo lo que me dieron. A mi hermano Mirko, por colaborar conmigo en las pequeñas cosas que hicieron que todo sea más fácil cuando el tiempo apremiaba.

A Mica, mi novia, que me bancó desde el primer día, soportando mis mañas, entendiendo el tiempo que demandaba una carrera universitaria como ésta y aguantándome en todos los momentos difíciles. Gracias por estar ahí y acompañarme en este momento tan importante.

A la familia de mi novia, en especial a Coco y Bochi, que siempre me empujaron para adelante y me hicieron sentir como uno más de la familia.

Por último, a mis amigos de la vida: Diego, Pablo, Matías, Damián y muchos otros que, con salidas y vacaciones, me sacaron de la rutina e hicieron toda esta etapa mucho más descontracturada.

Agradecimientos de Gabriela

Quiero agradecerle más que nada a toda mi familia. A mi mamá Diana y a mi papá Juan Carlos, que me incentivaron para que estudie una carrera universitaria, y me apoyaron durante todos estos años. A mi hermana Valeria, que estuvo al lado mío dándome fuerzas todo este tiempo.

Quiero agradecerle también a todas las personas especiales que estuvieron en mi vida a

lo largo de estos años de facultad, alentándome y acompañándome cada uno a su manera. A mis amigas del colegio, que me bancaron en los momentos lindos, y en aquellos no tan lindos también. A esos amigos que me esperaban para jugar al TEG cuando terminaba la época de parciales. A los chicos de teatro que se sumaron a mi familia, e hicieron que estos últimos años fueran mucho más divertidos. Y gracias también a todos los que no forman parte de ninguno de esos grupos y me apoyaron durante esta etapa.

Índice general

Capítulo 1

Introducción

Los problemas de *ruteo de vehículos* y *asignación de conductores* son dos de los principales problemas que surgen en las áreas de transporte, distribución y logística. El problema de ruteo de vehículos (VRP por su nombre en inglés, *Vehicle Routing Problem*)[TV02][Ge08] consiste en distribuir un conjunto de tareas entre un conjunto de vehículos disponibles, de forma de maximizar ganancias o minimizar costos de operación. Estas tareas pueden consistir en transportar pasajeros a lo largo de un segmento entre dos estaciones, llevar una carga de mercadería de un depósito a otro, etc.

En general, cuando se debe resolver un problema de ruteo de vehículos, surge también la necesidad de asignar conductores a cada una de las tareas que llevarán a cabo los vehículos. Este problema se conoce en la literatura como el problema de asignación de conductores o *Crew Scheduling Problem*[Ste07][Hui04]. Se entiende como *schedule* a la planificación de la jornada laboral para cada uno de los conductores, que usualmente está sujeta a regulaciones laborales. Algunos ejemplos de regulaciones son: la duración máxima de un turno de trabajo (horas consecutivas trabajadas), un día de descanso cada tantos trabajados, etc.

En la actualidad, muchas empresas a nivel mundial han adoptado paquetes de software comerciales para asistir en la logística de transporte, ya sea de mercadería como de pasajeros. Esto se debe principalmente a que:

- La desregulación del mercado incrementa la competencia entre las empresas por lo que optimizar el uso de cada recurso disponible es cada vez más importante.
- Las redes de transporte han crecido mucho en tamaño y cada vez es más difícil realizar una planificación razonablemente eficiente en forma manual.
- Las regulaciones laborales son complejas y muy dependientes de cada área particular: transporte urbano de pasajeros, transporte aéreo de pasajeros, transporte de mercadería, etc.

El VRP y el CSP son problemas clásicos en el área de Investigación Operativa y, como casi todos los problemas que trata este área, son computacionalmente difíciles de resolver. A pesar de esto, ha habido un esfuerzo importante en mejorar los procesos de optimización para obtener soluciones cada vez más eficientes. A su vez, la diversificación en el área de transporte ha hecho surgir nuevos problemas con características muy particulares. Esto hace que cada problema deba ser estudiado en forma independiente y requiera de una solución “a medida”.

1.1. Problemas de la literatura

En la literatura, el enfoque clásico para resolver problemas que involucran ruteo de vehículos y *scheduling* de conductores es *secuencial*. Esto quiere decir que primero se define qué vehículo realizará cada tarea, y después se distribuye la tripulación entre los vehículos. Este enfoque puede resultar útil para obtener soluciones factibles al problema, aunque al no considerar ambos problemas en forma simultánea, no hay forma de garantizar que la solución obtenida sea óptima. En muchos casos, la solución provista por el enfoque secuencial puede estar lejos del óptimo de una solución integrada [Ste07][Hui04]. Es por esto que muchos autores, como por ejemplo Friberg y Haase [FH96], o Freling, Huisman y Wagelmans [FHW03] proponen modelos que tratan ambos problemas de forma simultánea, definiendo un nuevo problema denominado VCSP (*Vehicle and Crew Scheduling Problem*).

El VCSP es utilizado principalmente para modelar el sistema de transporte público urbano. Es decir, dado un conjunto de viajes, se debe planificar qué colectivo realizará cada viaje, y qué chofer lo conducirá en cada segmento del mismo. Los datos que se conocen de cada viaje son el lugar y horario de partida, así como el lugar y horario de llegada. Para que un *schedule* sea solución factible del VCSP, es necesario que cada viaje se realice exactamente en el horario establecido.

Sin embargo, existen situaciones en las cuales se desea que los horarios sean flexibles dentro de cierto rango. Esas situaciones generalmente no se dan cuando se planifican sistemas de transporte de pasajeros, sino cuando se realiza distribución de mercadería. En esos casos, suele ocurrir que la mercadería pueda ser cargada y descargada en cualquier momento dentro de cierto rango horario o *ventana de tiempo*. Este problema es conocido en la literatura como VRPTW (*Vehicle Routing Problem with Time Windows*), y si bien resuelve la organización de los vehículos, no considera a los conductores.

Tanto el VRP como el VRPTW y el VCSP consideran que para satisfacer los pedidos de los clientes se debe transportar cierta mercadería desde un depósito único hasta la ubicación de cada cliente. Sin embargo, en algunos contextos, los pedidos de los clientes no sólo indican hacia dónde trasladar la mercadería, sino también en qué ubicación se encuentra ésta originalmente. Es decir, para cada pedido de un cliente, se especifica dónde debe recogerse la mercadería (es decir, donde se realiza el *pick-up*), y dónde debe entregarse (es decir, donde se realiza el *delivery* ó *drop-off*). A este problema se lo conoce como PDP (*Pickup and Delivery Problem*). Cuando en lugar de horarios fijos se establecen ventanas de tiempo para realizar los *pick-ups* y *drop-offs*, el problema se denomina PDPTW (*Pick-up and Delivery Problem with Time Windows*).

El problema que se aborda en esta tesis contiene elementos tanto del VCSP como del PDPTW, pero tiene además algunas características particulares que, a nuestro conocimiento, no tienen antecedente de haber sido estudiadas. A continuación presentamos una descripción del problema que trateremos.

1.2. Nuestro problema: *Vehicle and Crew Scheduling for Pickup and Delivery Problem with Time Windows* (VCSPDPTW)

Si bien la forma general del problema de distribución de mercadería está muy estudiada, dicha formulación general no siempre es aplicable en la práctica. Esto se debe a que de acuerdo al contexto en el cual se plantea el problema, se deben tener en cuenta características muy particulares. Un ejemplo de esto es que las restricciones que provienen

de regulaciones laborales deben ser formuladas según las leyes de cada país. Otro factor que puede modificar notablemente el problema es el de la capacidad de los vehículos a utilizar: a veces se asume que todos pueden transportar la misma cantidad de mercadería, pero otras veces no es posible basarse en esa hipótesis.

En esta tesis se aborda un problema de planificación de rutas para vehículos y asignación de conductores con características muy particulares. El problema está inspirado en un problema real de una compañía colombiana productora de café, y consiste en planificar el retiro y entrega de un conjunto de *pedidos*, cuya mercadería debe trasladarse desde un depósito en una determinada ciudad a otro depósito en otra ciudad. El retiro y entrega de cada pedido es realizado por alguno de los vehículos de los que dispone la empresa y, a su vez, cada vehículo es conducido por una tripulación a cargo. La tripulación está compuesta por uno o dos camioneros y puede variar a lo largo del recorrido de un vehículo. Es así como un camionero puede comenzar conduciendo un vehículo, realizar el retiro de un pedido, luego bajarse de dicho vehículo y subirse a otro. A su vez, el trabajo de los camioneros está sujeto a un conjunto de restricciones laborales que impiden, por ejemplo, que un camionero trabaje muchas horas consecutivas sin descanso.

El objetivo detrás de la resolución del problema consiste en planificar el transporte y entrega de toda la mercadería de forma de minimizar los costos operativos y, al mismo tiempo, garantizar que se respeten las restricciones laborales para los conductores de los vehículos. A continuación presentamos las características del VCSPDPTW.

1.2.1. Mapa de ciudades

El problema se define sobre un mapa de ciudades donde se conoce el tiempo que demora el viaje entre cada par de ciudades. Cada ciudad representa una ubicación en donde un vehículo o camión puede detenerse, ya sea para cargar mercadería de un depósito (realizar un *pick-up*), descargar mercadería en un depósito (realizar un *drop-off*), detenerse para descansar, realizar un intercambio de tripulación, etc. Sin perdida de generalidad se puede asumir que en cada ciudad hay a lo sumo un depósito, ya que si hubiera más, podría considerarse que se cuenta con varias ciudades distintas muy cercanas entre sí.

1.2.2. Camiones

Se cuenta con una cantidad fija de camiones, y para cada uno de ellos se conoce la ciudad donde se encuentra inicialmente. Se asume que los camiones viajan a una velocidad constante.

Cada camión puede transportar la carga correspondiente a un único pedido por vez. Es decir, si un camión recogió la mercadería de cierto pedido pero aún no la entregó, no puede realizar el *pick-up* de otro pedido. A su vez, cada carga de mercadería no ocupa más de un camión, por lo que la capacidad de los camiones no es relevante para el problema. De esta manera, cualquier camión puede llevar a cabo cualquier pedido.

Los camiones sólo pueden detenerse en las ciudades definidas en el mapa y sólo para cargar/descargar mercadería, para realizar un intercambio en su tripulación, o para que alguno de sus conductores descanse. La tripulación de un camión puede estar compuesta por uno o, a lo sumo, dos camioneros. La misma no es fija y puede variar a lo largo del recorrido, pero los cambios en la tripulación sólo pueden realizarse cuando el camión se detiene en una ciudad. Por ejemplo, un camión puede partir de una ciudad con un único camionero a bordo y luego llegar a otra ciudad donde asciende un segundo camionero. Más adelante en el recorrido pueden descender ambos del camión para que comience a

manejarlo un tercer camionero.

1.2.3. Camioneros

Se conoce la cantidad de camioneros disponibles, y en qué ciudad se encuentra cada uno en el momento inicial. Los camioneros pueden ascender y descender de los camiones sólo en las ciudades establecidas en el mapa. Para trasladarse entre ciudades, los camioneros utilizan los vehículos de la empresa y no pueden movilizarse utilizando otro medio de transporte.[1]

Un camionero no debe trabajar más de 12 horas en cualquier período de 24 horas. Notemos que esto no es exactamente lo mismo que exigir que se trabajen a lo sumo 12 horas por día, ya que si expresáramos la condición de esta manera, un camionero estaría habilitado para trabajar más de 12 horas consecutivas. Por ejemplo, si un camionero trabajara las últimas 10 horas de un día y las primeras 6 horas del día siguiente, cumpliría la limitación diaria pero habría trabajado durante 16 horas consecutivas.

Los camioneros deben tener al menos un día franco en cualquier período de 7 días. Se considera un día franco a aquel día en el que un camionero descansa durante toda la duración del mismo.

Se considera como tiempo de trabajo tanto al tiempo que pasa un camionero viajando en un camión como al tiempo que pasa cargando o descargando mercadería del mismo. Dado que el VCSPDPTW no considera el volumen de mercadería a transportar, se asume que el tiempo de carga y de descarga es siempre de 1 hora. Si un camión lleva dos camioneros a bordo, se computará el tiempo de viaje como tiempo trabajado para ambos, aunque uno sólo sea el que conduzca el camión. Si un camionero no está conduciendo un camión o acompañando al conductor, se considera que está descansando.

1.2.4. Pedidos

Un pedido de mercadería tiene una ciudad de origen, una ciudad de destino y fechas con rangos horarios de retiro y entrega.[2]

Las ciudades de origen y destino indican en qué ciudad se debe realizar el *pick-up* y el *drop-off* de la mercadería respectivamente. A su vez, cada pedido especifica la fecha a partir de la cual se puede realizar el *pick-up* y la fecha a partir de la cual se puede realizar el *drop-off*. La mercadería puede ser retirada a partir de la fecha especificada pero no es obligatorio hacerlo ese mismo día. Tampoco es obligatorio entregar el pedido en fecha, pero la entrega posterior tiene una penalidad asociada por cada día de demora. Esto no implica que el pedido pueda no ser realizado: todos los pedidos deben ser completados sin importar la fecha en la que se realiza la entrega.

Adicionalmente, cada pedido tiene rangos horarios o ventanas de tiempo en las que se permite cargar y descargar mercadería de los correspondientes depósitos. Estas ventanas de tiempo se repiten diaramente en forma fija, es decir, sin variar de acuerdo al día de la semana. Por ejemplo, si la ventana de tiempo para el *pick-up* de un pedido es de 14 a 16 hs. y el camión que retira la mercadería llega a las 16.30 hs., deberá esperar hasta el comienzo de la ventana del siguiente día para poder realizar el *pick-up*.

[1] Esta restricción es muy particular del VCSPDPTW. En el CSP un camionero puede finalizar su jornada en cierto lugar y comenzar la jornada siguiente en un lugar distinto, habiéndose trasladado allí por sus propios medios.

[2] A diferencia del VRPTW, la mercadería no se encuentra inicialmente en un único depósito sino que cada pedido tiene un depósito de origen particular.

1.2.5. Costo

El costo que debe minimizarse está compuesto por los siguientes factores:

- **Distancia total:** es sabido que cada kilómetro recorrido tiene un costo dado (influenciado principalmente por el costo del combustible), por lo cual debe tenerse en cuenta la distancia recorrida por cada camión en el costo total.
- **Penalidad por entrega fuera de fecha:** si bien cada pedido tiene una fecha recomendada de entrega, no es necesario realizar la entrega ese día. Los *drop-offs* (al igual que los *pick-ups*) pueden ser realizados cualquier día posterior al día especificado, pero cada día de demora en la entrega conlleva una penalidad que forma parte del costo total. Para cada pedido la penalidad diaria puede ser distinta, lo cual permite asignar prioridades entre los distintos pedidos.
- **Penalidad por viajar con camión sin mercadería:** dado que la finalidad de los camiones es transportar mercadería, cuando un camión viaja entre ciudades sin trasladar mercadería, se incrementa el costo de viaje. Esta penalidad es fija por kilómetro recorrido.

Los camioneros perciben un salario fijo, por lo que la cantidad de horas que trabajan no influye en el costo a minimizar.

Teniendo en cuenta todo lo descripto, se busca determinar qué camión estará a cargo de realizar cada pedido, cuando lo realizará y cómo serán los *schedules* de cada uno de los camioneros de forma de que el costo sea mínimo.

1.3. Resumen de la tesis

La propuesta de esta tesis es resolver el VCSPDPTW utilizando un modelo de programación lineal entera en conjunto con la técnica de generación de columnas. Este enfoque particiona al problema en dos: un problema maestro, que consiste en el modelo propuesto, y un subproblema de generación de columnas para dicho modelo. El problema maestro es formulado mediante un modelo de programación lineal entera binaria en donde las columnas a generar se corresponden con los posibles recorridos o *rutas* que pueden realizar los vehículos.

Para modelar el subproblema de generación de rutas utilizamos una *red con recursos y restricciones*. Un camino dirigido en esta red representa el recorrido que puede realizar alguno de los vehículos disponibles. El problema de generación de una ruta de costo óptimo se traduce entonces en encontrar un camino válido de costo mínimo en esta red. Para resolver esto diseñamos un algoritmo basado en la técnica de programación dinámica que construye caminos válidos óptimos.

Por otra parte, el problema maestro es resuelto mediante un algoritmo *Branch and Cut*, en donde la solución inicial requerida es provista por un algoritmo que genera un conjunto de rutas que satisface todos los pedidos incluidos en la definición del problema.

Realizamos pruebas de los algoritmos implementados sobre instancias de pruebas de distintos tamaños. Sobre las instancias más chicas logramos obtener buenos resultados en la mayoría de los casos. En cambio, sobre las instancias de mayor tamaño, la tasa de éxito fue menor que en las pruebas sobre instancias más chicas.

En el capítulo 2 presentamos algunos conceptos fundamentales de programación lineal entera y de teoría de grafos que son necesarios para comprender mejor el resto del

trabajo. El capítulo 3 presenta la formulación del modelo de programación lineal entera que utilizamos para resolver el problema. En el capítulo 4 definimos la red con recursos y restricciones que se utiliza para formalizar el concepto de ruta, es decir, de un recorrido válido que puede ser realizado por un vehículo. En el capítulo 5 describimos el método para generar las columnas a partir de la red definida anteriormente y de qué manera se introducen estas columnas en el modelo presentado en el capítulo 3. El capítulo 6 presenta el esquema algorítmico de resolución general del problema, integrando la resolución del problema maestro con el subproblema de generación de columnas. En el capítulo 7 se muestran las pruebas que realizamos y los resultados que obtuvimos. Finalmente, en el capítulo 8 presentamos las conclusiones del trabajo así como el trabajo a futuro que podría realizarse.

Capítulo 2

Preliminares

En un problema de optimización combinatoria, se desea encontrar una solución óptima dentro de cierto conjunto finito de soluciones factibles. Usualmente este conjunto de soluciones es tan grande que no es posible realizar una búsqueda exhaustiva sobre el mismo de manera de encontrar la mejor solución. Este tipo de problemas son de gran relevancia en la práctica y surgen en casi todas las áreas de management (e.g. finanzas, marketing, producción, etc) así como en muchas áreas de ingeniería (e.g. diseño de circuitos, diseño de redes de telecomunicaciones, logística para el transporte de energía eléctrica, logística para el transporte de pasajeros, etc). Algunos de los problemas clásicos de optimización combinatoria son los siguientes: *Travelling Salesman Problem*, *Knapsack Problem*, *Set Covering Problem*, etc [Wol98].

Muchos problemas de optimización combinatoria pueden ser tratados con modelos de programación lineal entera y, en particular, binaria. Generalmente un modelo de programación lineal entera es fácil de formular y, en muchos casos, existen varias formulaciones diferentes para un mismo problema. La dificultad se presenta a la hora de la resolución de dichas formulaciones.

El objetivo de este capítulo es describir los conceptos fundamentales y algoritmos de resolución para problemas de programación lineal y programación lineal entera. No pretende ser un compendio completo acerca del tema sino una descripción breve y concisa de manera que el lector pueda comprender mejor los siguientes capítulos.

2.1. Programación Lineal

La programación lineal permite expresar un problema en términos matemáticos para determinar cuál es la mejor solución dentro de un conjunto, generalmente grande, de soluciones factibles.

Más formalmente, la programación lineal es una técnica para optimizar una función lineal, conocida como *función objetivo*, sujeto a que la solución de dicha optimización satisfaga un conjunto de igualdades y desigualdades lineales, conocidas como *restricciones lineales*.

Definición 2.1 (Problema de programación lineal)
Un problema de programación lineal (PPL) *se expresa en forma canónica como:*

$$\begin{array}{ll} \text{maximizar} & c^t x \\ \text{sujeto a} & Ax \leq b \\ & x \geq 0 \end{array}$$

Donde:

- $x \in \mathbb{R}^n$ *representa el vector de variables positivas cuyo valor se debe determinar,*
- $c \in \mathbb{R}^n$ *es el vector de coeficientes de la función objetivo,*
- $b \in \mathbb{R}^m$ *es el vector de términos independientes,*
- $A \in \mathbb{R}^{m \times n}$ *es la matriz de coeficientes de las restricciones.*

Un PPL también puede notarse como $z^* = max\{cx : Ax \leq b, x \geq 0\}$*, donde* z^* *es el valor óptimo de la función objetivo.*

Las soluciones factibles de un PPL están caracterizadas por sus restricciones lineales. A la región definida por ellas se la conoce como *poliedro*.

Definición 2.2 (Poliedro)
Dados $A \in \mathbb{R}^{m \times n}$ *y* $b \in \mathbb{R}^m$*, un subconjunto de* $\mathbb{R}^n$ *descripto por un conjunto finito de restricciones lineales* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ *es un* poliedro.

Si bien en 2.1 se plantea un PPL en su forma estándar, puede ser necesario, dada la naturaleza del problema a resolver, plantearlo de una manera diferente. Por ejemplo: como un problema de minimización en vez de uno de maximización, con igualdades como restricciones en vez de desigualdades, etc. Lo importante es que cualquiera sea la variante del PPL, existe otro equivalente en su forma canónica.

Los problemas en donde el dominio de las variables es continuo (variables reales) pueden ser resueltos con el método *Simplex*[1]. Este método se basa en el hecho de que la solución óptima de un PPL es alguno de los puntos que delimitan al poliedro del mismo, conocidos como *puntos extremos*. De esta manera, *Simplex* explora los puntos extremos del poliedro en forma *inteligente*, eligiendo en cada iteración un punto que aumente el valor de la función objetivo (en el caso de la maximización). Este proceso se realiza en forma iterativa hasta que el valor de la función objetivo no pueda aumentar más: esto significa que se alcanzó el óptimo.

Está demostrado que el método *Simplex* tiene una complejidad algorítmica exponencial en el peor caso. Sin embargo, y a pesar de existir algoritmos polinomiales para resolver PPLs (por ejemplo, el presentado en [Kar84]), *Simplex* demostró tener un muy buen desempeño en la práctica. Esto lo convirtió en uno de los métodos de resolución de PPLs más utilizados en la actualidad.

2.2. Programación lineal entera

Muchos de los problemas que se presentan en la práctica y que pueden ser planteados como un PPL presentan la dificultad adicional de que el dominio de las variables no es continuo sino discreto. Por ejemplo, si las variables sólo pueden tomar valores enteros, estamos ante un *Problema de Programación Lineal Entera* (PPLE); si algunas variables pueden tomar valores enteros y otras valores reales, estamos ante un *Problema de Programación Lineal Entera Mixto* (PPLEM) y finalmente si las variables sólo pueden tomar valores 0 o 1, estamos ante un *Problema de Programación Lineal Entera Binario* (PPLEB).

[1] Método de resolución general para problemas de programación lineal desarrollado por George Dantzig. Para una descripción detallada del método ver [Chv83].

Un problema de programación lineal entera binario sirve, en general, para modelar situaciones donde deben tomarse decisiones por *sí* o por *no* en forma óptima. Algunos ejemplos de estos problemas son: la inversión en un portfolio de acciones, la planificación de vuelos para una flota de línea aérea, la planificación en la generación de electricidad, etc. Resulta muy natural en estos casos asociar a cada decisión una variable x restringida a tomar valores 1 o 0 que representen la respuesta por *sí* o por *no* respectivamente.

Formalmente, un problema de programación lineal entera binaria busca el óptimo de una función lineal entre los vectores binarios que se encuentran dentro del poliedro caracterizado por las inecuaciones y ecuaciones lineales definidas por A. La formulación es idéntica a la presentada en 2.1 excepto que $x \in \{0,1\}^n$. Un PPLEB es un caso particular de un problema de programación lineal entera, por lo que si mencionamos propiedades sobre estos últimos se debe entender que también son aplicables a los primeros.

La diferencia fundamental entre los problemas donde el dominio de las variables es discreto y aquellos con variables continuas es que los primeros pertenecen a la clase *NP-hard* [Pap81], por lo que no se conoce un algoritmo polinomial que los resuelva en forma general. Al no contar con un método general que pueda tratar con estos problemas en forma eficiente, se suele usar la idea de *relajación* del problema. Esto consiste en reemplazar el problema original por uno similar que contemple todas las soluciones del anterior y que, al mismo tiempo, pueda resolverse más fácilmente.

Definición 2.3 (Relajación de un PPL)
Un problema $PR \equiv max\{f(x) : x \in T \subseteq \mathbb{R}^n\}$ *es una* relajación *de* $PE \equiv max\{c(x) : x \in X \subseteq \mathbb{R}^n\}$ *si:*

- $X \subseteq T$, *y*
- $f(x) \geq c(x)$ *para todo* $x \in X$.

Una manera bastante sencilla de relajar un PPLE es eliminar la condición de que las variables tomen valores enteros y permitir que tomen valores reales. Esta relajación es conocida como *Relajación Lineal* y el problema relajado termina siendo un PPL. Un enfoque *naive* para obtener una solución de un PPLE podría ser resolver la relajación lineal y luego redondear la solución obtenida de manera que termine siendo una solución factible de problema original. Si bien esto parece una buena heurística que se podría aplicar en forma general, existen muchos ejemplos en los cuales una solución redondeada dista mucho de la solución óptima del problema original. Para los PPLEB la situación es inclusive peor: una solución de la relajación bien podría ser (0,5, ..., 0,5), en cuyo caso no sabríamos si redondear hacia el 0 o el 1.

A pesar de la dificultad que presenta la resolución en forma general de PPLEs, existen algunos casos que pueden ser resueltos en forma eficiente. Llamemos S al conjunto de soluciones factibles binarias del problema, es decir $S = \{x \in \{0,1\}^n : Ax \leq b\}$ y P a la cápsula convexa de S (menor poliedro que contiene a S). Si P coincide con el poliedro asociado a la relajación lineal de S, $S_{RL} = \{x \in \mathbb{R}^n, 0 \leq x \leq 1 : Ax \leq b\}$, basta con resolver la relajación lineal del problema para obtener el óptimo buscado.

Lamentablemente, para la mayoría de los problemas no se conoce la caracterización de su cápsula convexa y el esfuerzo que se requiere para caracterizarla puede resultar tan difícil como resolver el problema original. En estos casos, la relajación lineal no resuelve el problema en forma completa pero puede brindarnos cierta información útil.

Observación 2.1 *Si una relajación de un PPLE es un problema infactible (i.e. no tiene solución factible) el PPLE también lo es.*

Observación 2.2 *Si una relajación de un PPLE es un problema factible, el valor óptimo de la relajación es una cota superior (caso maximización) o inferior (caso minimización) del valor óptimo del PPLE.*

Observación 2.3 *Si una relajación de un PPLE tiene una solución factible óptima $x^* \in \mathbb{Z}^n$ entonces x^* es solución óptima para el PPLE.*

La diferencia entre el valor óptimo del PPLE y el de su relajación lineal se conoce como *gap* y suele usarse como medida de la calidad de la relajación. Como no es posible conocer a priori el valor del *gap* se utiliza una cota superior del mismo dada por la diferencia entre el valor de una solución factible del PPLE y el valor óptimo de la relajación. Esto permite conocer la calidad de una solución y, en algunos casos, certificar la optimalidad de la misma (cuando ambos valores coinciden).

Muchos de los algoritmos para resolver problemas de programación lineal entera se basan en estas propiedades que relacionan un PPLE y su relajación lineal. A continuación describiremos algunos de los más utilizados en la práctica.

2.3. Algoritmos para Problemas de Programación Lineal Entera

La mayoría de los métodos de resolución exacta para modelos de programación lineal entera se encuadran en alguno de los siguientes esquemas:

- Métodos de Planos de corte
- Métodos *Branch and Bound*
- Métodos *Branch and Cut*
- Métodos *Branch and Price*

2.3.1. Métodos de Planos de Corte

La idea fundamental de un algoritmo de planos de corte es considerar la relajación lineal de un problema con soluciones factibles S e ir mejorándola con el agregado de *desigualdades lineales válidas* de manera que se eliminen o *corten* soluciones del poliedro S_{RL} (asociado a la relajación) que no pertenezcan a S. Su objetivo es ir refinando el poliedro S_{RL} para acercarse lo más posible a la cápsula convexa de S y, eventualmente, encontrar una solución entera al problema.

El esquema general del algoritmo comienza resolviendo la relajación del problema omitiendo las condiciones de que las variables sean enteras. Si al menos una variable que debe ser entera resulta no serlo, se busca identificar una desigualdad lineal válida para el poliedro que *separe* la actual solución del conjunto de soluciones factibles enteras. Al agregar esta desigualdad a la formulación se obtiene una nueva relajación del problema más ajustada, sobre la cual puede repetirse este procedimiento. El éxito de la metodología depende en gran medida de la posibilidad y la eficiencia de encontrar desigualdades violadas, llamadas *planos de corte*, que puedan ser agregadas a la formulación para separar las soluciones que no sean enteras. En el algoritmo 1 presentamos el esquema básico de planos de corte.

Los planos de corte pueden ser generados bajo dos enfoques: con herramientas generales aplicables a cualquier PPLE o explotando las particularidades de cada problema.

Algoritmo 1 Planos de corte

Input: $PE \equiv max\{f(x) : x \in S\}$
Output: x^* solución óptima, o informa que no pudo encontrarla

1: *// Resolver relajación lineal*
2: Resolver $PR \equiv max\{f(x) : x \in S_{RL}\}$ la relajación lineal del problema. Sea x^* su solución óptima
3: **if** x^* es solución entera **then**
4: **return** x^*
5: **end if**
6: *// Separación*
7: Buscar un conjunto D de desigualdades válidas violadas por x^*
8: **if** $D \neq \emptyset$ **then**
9: Agregar las desigualdades de D a S_{RL}
10: Volver al paso 2
11: **else**
12: **return** No se pudo encontrar solución
13: **end if**

El primer enfoque fue desarrollado a comienzos de los años 60 por Gomory [Gom58] y permite generar desigualdades válidas que cortan la solución de la relajación. La ventaja de este método es que es de aplicación general, es decir, no utiliza información específica acerca del problema y, bajo ciertas condiciones, es convergente. Este tipo de cortes están implementados en la mayoría de los paquetes de optimización como CPLEX® [IBM10].

El segundo enfoque se basa en explotar la información particular del problema a resolver para identificar planos de corte que permitan separar la mayor cantidad de soluciones no enteras. Para generar este tipo de desigualdades es necesario realizar un *estudio poliedral* del problema, que consiste en tratar de describir la cápsula convexa de las soluciones factibles del mismo. Los primeros estudios poliedrales fueron realizados a principios de la década de los 70, para el *Maximum Set Packing Problem* [Pad73] y el *Travelling Salesman Problem* [GP79] y resultaron un gran avance en la resolución de este tipo de problemas.

2.3.2. Algoritmos *Branch and Bound*

El método *Branch and Bound* se enmarca en la técnica algorítmica de *Divide and Conquer*. Esta técnica consiste en dividir un problema en una serie de problemas más chicos, por ende más fáciles de resolver, y luego componer las soluciones de los mismos para resolver el problema original.

Consideremos el problema $z^* = max\{cx : x \in S\}$. Sea $S = S_1 \cup \ldots \cup S_q$ una descomposición del espacio de soluciones factibles S en subespacios más chicos y $z^k = max\{cx : x \in S_k\}$ para $k = 1, \ldots, q$. Se verifica entonces que $z^* = max_k\ z^k$, lo que establece cómo componer las soluciones de los subproblemas para obtener una solución del problema original.

La forma típica de descomposición es dividir el espacio de soluciones factibles en subconjuntos en forma recursiva. Este proceso de división se conoce como *branching* y permite generar un *árbol* cuya raíz corresponde al conjunto S original, y sus nodos corresponden a los diferentes subconjuntos $S_1, \ldots, S_q$ resultantes de la descomposición. Luego, cada subconjunto S_k es nuevamente particionado en otros subconjuntos aplicando el *branching* en forma recursiva. De esta manera, la resolución de los subproblemas asociados a los nodos del árbol contribuirán a la resolución del problema asociado al nodo raíz.

Debido al tamaño que puede alcanzar el árbol, es esencial disponer de alguna forma de evitar explorar ramas completas, es decir, disponer de algún mecanismo de *poda*. Este mecanismo de poda se basa en información parcial provista por la resolución de los subproblemas. Más especificamente, depende de las cotas (*bounds*) inferiores y superiores para el valor óptimo z^* que son obtenidas de dichos subproblemas.

La implementación más difundida y utilizada en la mayoría de los paquetes comerciales de optimización utiliza la relajación lineal para el proceso de *branching* y fundamentalmente para obtener las cotas que permiten podar el árbol. A cada nodo del árbol se le asocia la relajación lineal del problema en el subespacio de búsqueda correspondiente a ese nodo. En el caso de que la solución del problema relajado sea entera, ésta es la solución buscada en esa región del espacio. Si el subproblema relajado no tiene solución o su valor óptimo (cota superior para el óptimo entero) es peor que la mejor solución entera conocida hasta el momento (mejor cota inferior), no hay necesidad de seguir explorando y dividiendo el subconjunto de soluciones asociado al nodo. Por lo tanto, en cualquiera de los casos mencionados la rama del árbol que se genera a partir del nodo puede ser podada. En cambio, si al menos una variable de la solución del problema relajado no es entera y el valor óptimo de la relajación es mejor que la mejor solución entera que se dispone, no hay razón para detener la búsqueda en esa región del espacio. En este caso se debe realizar un nuevo *branching* para generar nuevos nodos y así continuar con la exploración.

En el algoritmo 2 presentamos el esquema básico del *Branch and Bound*. En este esquema no está especificada la regla a seguir para la elección de un nodo de la lista ni el proceso de generación de los subproblemas. Respecto a la elección de un nodo, las opciones más usuales son: DFS (*Depth First Search*, último nodo de la lista), BFS (*Breadth First Search*, primer nodo de la lista) o BeFS (*Best First Search* nodo con mejor valor óptimo de la relajación).

Algoritmo 2 *Branch and Bound*

Input: $PE \equiv max\{f(x) : x \in S\}$
Output: $\overline{x}$ solución óptima

1: // *Inicialización*
2: Sean $L \equiv \{PR\}$ donde PR es la r.l. de PE, $\underline{z} = -\infty$ una cota inferior y $\overline{x}$ la mejor sol. entera encontrada hasta el momento
3: // *Elección de nodo*
4: **if** $L = \emptyset$ **then**
5: **return** $\underline{z}$
6: **end if**
7: Elegir una r.l. $R \in L$, quitarla de L y resolverla
8: **if** R no resulta factible **then**
9: Volver al paso 4
10: **end if**
11: // *Bound*
12: Sean x^* la solución óptima de R y z^* el valor de la función objetivo
13: **if** x^* es solución entera **then**
14: $\underline{z} \leftarrow max(\underline{z}, z^*)$
15: Actualizar $\overline{x}$ con x^* si corresponde
16: **else if** $z^* > \underline{z}$ **then**
17: // *Branch*
18: Generar subproblemas de R y agregarlos a la lista L
19: **end if**
20: Volver al paso 4

Para generar los subproblemas suele usarse la clásica dicotomía en una variable x_i. Del conjunto de variables no enteras de la solución óptima de la relajación, se puede elegir x_i como la variable tal que: x_i^* sea el valor más cercano a un número entero, i sea el menor índice de variable, x_i cumpla alguna propiedad específica del problema, etc. Los dos nuevos nodos que se generan tienen asociada la región del espacio del nodo padre con el agregado de las restricciones $x_i \leq \lfloor x_i^* \rfloor$ y $x_i \geq \lceil x_i^* \rceil$ respectivamente. Cualquier combinación de estas reglas da origen a un árbol distinto y, en general, no es posible determinar si es mejor aplicar una regla u otra.

2.3.3. Algoritmos *Branch and Cut*

El método de *Branch and Cut* surge a comienzos de los años 80 como un híbrido entre los algoritmos de *Branch and Bound* y planos de corte. La idea del método sigue los lineamientos generales de un algoritmo de *Branch and Bound* con la diferencia de que se aplican algoritmos de planos de corte sobre las relajaciones lineales asociadas a los nodos del árbol. El objetivo de la aplicación de cortes es obtener mejores cotas en las relajaciones lineales de cada nodo y, de esta forma, reducir considerablemente el tamaño del árbol.

Algoritmo 3 *Branch and Cut*

Input: $PE \equiv max\{f(x) : x \in S\}$
Output: $\overline{x}$ solución óptima

1: // *Inicialización*
2: Sean $L \equiv \{PR\}$ donde PR es la r.l. de PE, $\underline{z} = -\infty$ una cota inferior y $\overline{x}$ la mejor sol. entera encontrada hasta el momento
3: // *Elección de nodo*
4: **if** $L = \emptyset$ **then**
5: **return** $\underline{z}$
6: **end if**
7: Elegir una r.l. $R \in L$ y quitarla de L
8: Resolver R
9: **if** R no resulta factible **then**
10: Volver al paso 4
11: **end if**
12: // *Bound*
13: Sean x^* la solución óptima de R y z^* el valor de la función objetivo
14: **if** x^* es solución entera **then**
15: $\underline{z} \leftarrow max(\underline{z}, z^*)$
16: Actualizar $\overline{x}$ con x^* si corresponde
17: **else if** $z^* > \underline{z}$ **then**
18: // *Branch vs Cut*
19: **if** Corresponde agregar cortes **then**
20: Buscar un conjunto D de desigualdades válidas violadas por x^*
21: **if** $D \neq \emptyset$ **then**
22: Agregar las desigualdades de D a la formulación
23: Volver al paso 8
24: **end if**
25: **end if**
26: Generar subproblemas de R y agregarlos a la lista L
27: **end if**
28: Volver al paso 4

En el algoritmo 3 presentamos el esquema básico de un *Branch and Cut*. En la descripción del algoritmo quedan muchos puntos sin especificar. Por ejemplo, ¿con qué criterio decidir si se deben agregar cortes o no?, ¿cuántos cortes agregar?, ¿se deben aplicar cortes globales (válidos para cualquier subproblema) o cortes locales (válidos sólo en el subproblema actual)? La eficiencia del algoritmo depende de estos factores y de muchos otros. En la práctica, lograr un equilibrio entre ellos no es una tarea fácil y depende en gran medida del problema particular que se quiere resolver.

2.3.4. Generación de columnas y algoritmos *Branch and Price*

Muchas formulaciones de PPLs tienen la característica de contar con un gran número de variables. Usualmente este número es tan grande que no es posible considerar todas las variables en forma explícita. Este tipo de formulaciones no son una rareza y son usadas en la práctica por diversas razones. Puede ocurrir que sean los únicos modelos conocidos para un determinado problema y no exista un modelo alternativo. También puede suceder que modelos alternativos con menor cantidad de variables tengan relajaciones más débiles y no resulten buenos para ser utilizados con los algoritmos conocidos.

La estrategia, en estos casos, es resolver la relajación lineal restringiéndose sólo a un subconjunto de variables. Esto se basa en el hecho de que la mayoría de las variables no serán parte de la solución óptima, por lo que en teoría sólo es necesario un subconjunto chico de todas ellas para resolver el problema. Al no considerar todas las variables, la solución obtenida será factible pero no hay garantía de que sea óptima. Para cerciorarse de esto es necesario buscar entre las variables no consideradas si existe alguna candidata que pueda mejorar el valor de la función objetivo. Este proceso se denomina *generación de columnas*.

Más especificamente, la técnica de generación de columnas separa al problema original en dos: un problema *maestro* y un *subproblema de generación de columnas*. El problema maestro es el problema original pero sólo considerando un subconjunto de todas las variables. El subproblema de generación de columnas consiste en identificar una nueva variable (o columna) para agregar al subconjunto mencionado antes. La nueva columna a ingresar debe ser generada de tal manera que, de ser incluida en la base, mejore el valor de la función objetivo.

En el caso de una minimización, *Simplex* establece que para mejorar la función objetivo en una iteración dada, la nueva columna debe tener un *costo reducido negativo*. En términos matemáticos esto significa que:

$$c_i - y^* A_i < 0$$

donde A_i es la columna i de la matriz A, c_i es el coeficiente de costo de la variable correspondiente a la columna i y y^* es el vector solución del problema *dual* asociado al problema original restringido. Los valores de dicho vector se conocen como *prices* de las columnas de A y, de cierta manera, ponderan la decisión de qué variable debe ingresar a la base en la correspondiente iteración.

El subproblema se encargará entonces, en cada iteración del problema maestro, de buscar una columna A_i que verifique la anterior inecuación. Si dicha columna existe, se agregará al subconjunto de variables y se reoptimizará el problema maestro para luego repetir el proceso. Si por el contrario, la columna buscada no existe, no será posible mejorar el valor de la función objetivo. En este caso, se puede concluir que se alcanzó el óptimo del problema maestro.

En las formulaciones tradicionales, donde se cuenta con todas las variables en forma explícita, es fácil obtener una solución factible inicial. En cambio, en un modelo que uti-

liza generación de columnas, es necesario proveer un subconjunto inicial de variables que conformen una solución factible para el problema maestro. Dependiendo del problema, esto puede resultar trivial o puede requerir de un algoritmo complejo de generación de columnas iniciales.

Branch and Price

La técnica de generación de columnas tal como la describimos hasta ahora es aplicable a la resolución de problemas de programación lineal por el método *Simplex*. Cuando el problema a resolver es un PPLE, se combinan los métodos de generación de columnas con un esquema de *Branch and Bound* para dar lugar al método de *Branch and Price*. El esquema general del método es idéntico al del *Branch and Bound* original, con la diferencia de que la resolución de la relajación lineal asociada a cada nodo del árbol se realiza con un método de generación de columnas.

En el contexto de un algoritmo de *Branch and Price* es muy importante contar con un buen algoritmo de generación de columnas que permita resolver las relajaciones sin consumir mucho tiempo. Si bien esta condición es deseable, no siempre es posible contar con un algoritmo eficiente para resolver el subproblema en forma exacta. De hecho, no es raro que el subproblema termine siendo igual de difícil de resolver, en términos de complejidad algorítmica, que el problema original. En estos casos se suelen utilizar heurísticas que permitan obtener una columna con costo reducido negativo pero sin la garantía de que dicho costo sea el mínimo. Sin esta garantía ya no se puede asegurar, al no obtener una columna con costo reducido negativo, que se ha alcanzado el óptimo de la relajación. Cuando ocurre esto puede utilizarse un algoritmo exacto o simplemente finalizar la etapa de generación de columnas.

El principal problema que se genera al combinar los métodos de *Branch and Bound* y generación de columnas es que la estrategia de *branching* impacta directamente sobre la resolución del subproblema. Por ejemplo, supongamos que tenemos un PPLEB y decidimos realizar un *branching* por una determinada variable fraccionaria x_i. En una de las ramas resultantes tendremos la restricción extra $x_i = 0$ y en la otra $x_i = 1$. El algoritmo de generación de columnas deberá tener en cuenta la información de que la variable x_i no estará incluida en la solución o sí lo estará respectivamente. Esto podría modificar el subproblema de generación en cada nodo del árbol. El impacto de la modificación depende fuertemente de la estructura del subproblema y a veces puede resultar muy difícil de implementar. En estos casos se suele utilizar una solución de compromiso que consiste en generar columnas únicamente en el nodo raíz, parar el proceso de generación según algún criterio y luego continuar con un *Branch and Bound* o *Branch and Cut*. Por supuesto que al realizar esto se está sacrificando la garantía de optimalidad de la solución obteniéndose como resultado un algoritmo heurístico.

2.4. Teoría de Grafos

Las estructuras representadas por nodos y sus conexiones se presentan en una gran variedad de aplicaciones en la vida real. Algunos ejemplos de conexiones que tienen una representación física son los circuitos eléctricos, mapas de rutas y modelos moleculares. Otros se presentan en forma menos tangible, como pueden ser las relaciones sociales, un modelo de base de datos o el flujo de ejecución de un programa.

A continuación presentamos algunas nociones sobre teoría de grafos. Éstas ayudarán a comprender mejor el modelo utilizado en este trabajo para resolver el subproblema de

generación de columnas.

Definición 2.4 (Grafo)
Un grafo $G = \langle N, E \rangle$ *es una dupla donde* N *es un conjunto de vértices (o nodos) y* $E \subseteq N \times N$ *es un conjunto de aristas (o ejes) que representan la relación entre los elementos de* N*.*

En principio una arista $(a, b) \in E$ representa la relación entre dos nodos $a, b \in N$ y se dice que dichos nodos son *vecinos*. Las aristas pueden también tener una dirección en cuyo caso el conjunto E pasa a ser un conjunto de pares ordenados y cada arista tendrá un nodo de origen y un nodo de destino.

Definición 2.5 (Grafo dirigido)
Un grafo dirigido *o* digrafo $G = \langle N, E \rangle$ *es un grafo cuyas aristas son dirigidas, es decir, tienen un nodo origen y un nodo destino.*

Muchas aplicaciones que se pueden modelar utilizando grafos necesitan expresar la noción de *camino* entre dos vértices.

Definición 2.6 (Camino)
Dado un grafo o digrafo $G = \langle N, E \rangle$*, un* camino C *entre los vértices* $v_0, v_n \in N$ *es una secuencia de aristas con la siguiente forma*

$$C = [(v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_n)]$$

Si el grafo es dirigido, lo llamaremos camino dirigido.

También se pueden definir atributos sobre las aristas y vértices de un grafo. Uno de los atributos más utilizados sobre aristas es el de *costo* o *peso*. Por ejemplo, el costo sobre una arista podría representar tiempo de viaje o costo de transporte.

Definición 2.7 (Grafo con atributos)
Un grafo con atributos *es una tupla* $G = \langle N, E, f_1, f_2, ..., f_n \rangle$*, donde* N *y* E *son los conjuntos de vértices y aristas respectivamente y las* f_i *son funciones con dominio* N *o* E *y codominio* C_i*, donde* C_i *es un conjunto cualquiera.*

En el ejemplo anterior, un grafo con costo en sus aristas se podría definir como $G = \langle N, E, \sigma \rangle$ donde $\sigma : E \to \mathbb{R}$ y representa el costo de cada arista. Este tipo de grafos son muy utilizados en problemas de optimización combinatoria, en donde generalmente se busca un camino entre dos vértices tal que el costo de dicho camino, es decir la suma de los costos de sus aristas, sea mínimo. Este es un problema clásico de la teoría de grafos y se conoce como *problema de camino mínimo* (SPP por su nombre en inglés *Shortest Path Problem*). Puede ser resuelto en forma eficiente con algoritmos como el de Dijkstra [Dij59] o el de Floyd [Flo62].

2.4.1. Redes

Una *red* es un tipo particular de grafo dirigido. En general son utilizadas para modelar redes físicas como pueden ser redes de comunicación u oleoductos pero también se utilizan, por ejemplo, como modelo de generación de rutas de vehículos que deben ir de un lugar determinado a otro. Este último caso es el que resulta de particular interés para nuestro trabajo. A continuación definimos formalmente el concepto de red.

Definición 2.8 (Red)
Una red *es un grafo dirigido* $G = \langle N \cup \{s, t\}, E \rangle$*, donde* s *y* t *son dos vértices distinguidos, llamados* fuente *o* source *y* sumidero *o* sink *respectivamente, con la propiedad de que* s *es vértice origen de al menos una arista y* t *es destino de al menos una arista.*

En muchos de los problemas relacionados con la generación de rutas de vehículos y/o *schedules* de conductores se presenta, como modelo de generación subyacente, a una *red con restricciones sobre recursos*[FH96] [MP08]. La idea básica es que un camino entre los nodos s y t represente una ruta o *schedule* (dependiendo del problema) que sea factible o válido. La factibilidad del camino va a estar dada por los recursos y las restricciones que se aplicarán sobre ellos.

Un recurso corresponde a una cantidad, como por ejemplo el tiempo, el peso de la carga que lleva un vehículo o la duración de un descanso en un período de trabajo. Esta cantidad es acumulada a lo largo de un camino de acuerdo a lo que indique cada función asociada al recurso. Un recurso puede ser visto entonces como un atributo de la red en el sentido que las funciones que los caracterizan se definen sobre nodos o aristas y asignan un valor a cada uno de ellos.

Una restricción sobre un recurso está dada por una condición que debe verificar el recurso acumulado a lo largo del camino. Un ejemplo de restricción podría ser la condición de que el tiempo acumulado por un camino esté comprendido dentro de cierto intervalo o ventana. Una restricción puede ser aplicada sobre un nodo o una arista, y expresa cierta condición que debe cumplirse con respecto a la acumulación de un recurso por un camino hasta dicho nodo o arista. Remitiéndonos al ejemplo anterior, sea $[a_i, b_i]$ la ventana de tiempo asociada al nodo i, un camino será factible si el tiempo t en que visita dicho nodo verifica que $t \leq b_i$. La condición de que $t \geq a_i$ es opcional de acuerdo al problema que se esté tratando.

2.4.2. Camino mínimo con restricciones sobre recursos

El problema de camino mínimo con restricciones sobre recursos (SPPRC por su nombre en inglés *Shortest Path Problem with Resource Constraints*) fue presentado por primera vez por Desrochers [Des86] para resolver un problema de *scheduling* de camioneros. Consiste en encontrar un camino mínimo entre los nodos s y t de una red, tal que dicho camino satisfaga un conjunto de restricciones definidas sobre recursos.

A diferencia del problema de camino mínimo original, en el SPPRC dos caminos pueden ser *incomparables*. Esto sucede cuando un camino es mejor que otro de acuerdo a un criterio y peor de acuerdo a otro. Por ejemplo, si consideramos el costo y el tiempo de un camino como recursos, puede suceder que un camino parcial C_1 tenga costo menor a otro camino C_2, pero que C_2 tenga menor tiempo que C_1. En este caso no podemos decidir, a priori, cuál de los caminos conviene tomar. Las restricciones sobre los recursos fuerzan a que se consideren todos los caminos incomparables que alcanzan un determinado nodo. Esto es así dado que las restricciones podrían no permitir extensiones factibles de algunos de estos caminos pero sí de otros.

La particularidad de que los caminos se tengan que comparar en base a varios criterios hace que el SPPRC no pueda ser resuelto con algoritmos clásicos como el de Dijkstra. De hecho, está demostrado que el SPPRC que busca caminos elementales (i.e. sin vértices repetidos) es *NP-hard* [ENS08], y la variante que permite caminos con vértices repetidos puede ser resuelta en tiempo *pseudo-polinomial* [ID04].

El SPPRC se presenta como un mecanismo para generar columnas en muchos de los problemas de ruteo de vehículos y *scheduling* para conductores. Un camino mínimo repre-

sentará entonces una ruta o *schedule* óptimo, lo que garantizará una mejora en el valor de la función objetivo en la iteración correspondiente. El hecho de que en cada iteración se genere una columna por medio del SPPRC da la pauta de que el algoritmo que lo resuelva debe ser eficiente. Sin embargo, puede suceder que la red sobre la cual se busca el camino sea muy grande y/o que la comparación de criterios sea muy costosa. Esto puede implicar que la generación de columnas mediante un algoritmo exacto resulte prohibitiva. En estos casos se pueden relajar los criterios de comparación y en lugar de tener un algoritmo exacto tendremos una heurística.

Capítulo 3

Modelo

Resolver los problemas de VRP y CSP mediante métodos de programación lineal es de por sí una tarea difícil. El enfoque de esta tesis es resolver estos problema en forma conjunta, sin independizar la planificación de los vehículos de la de sus conductores. Esto sumado a las particularidades del problema tratado hizo que los modelos de programación lineal que analizamos tuvieran demasiadas variables para considerarlos en forma explícita.

La gran cantidad de variables que se requieren puede verse claramente cuando consideramos al tiempo como una de ellas. En nuestro problema es preciso conocer el instante de tiempo en que se realizan ciertas acciones, como ser: el ascenso y descenso de camioneros, el comienzo del descanso de un camionero, el *pick-up* de un pedido, etc. Si bien el tiempo puede discretizarse de varias maneras (días, horas, minutos, segundos), se debe optar por una discretización que sirva a los fines de la resolución práctica del problema, por ejemplo se puede dividir el día en 24 horas. Ésta y cualquier discretización más *granular* hace que el número de variables crezca enormemente al combinarse con los otros aspectos del problema que debe tener en cuenta el modelo (ciudades, camioneros, etc).

Por esta razón es que optamos por un modelo de generación de columnas para resolver el problema. A continuación se presentan los detalles del modelo propuesto para el problema maestro. En el capítulo 5 se presentará el modelo para el subproblema de generación de columnas, para de esta manera complementar el esquema general de resolución.

3.1. Definiciones preliminares

Antes de definir formalmente el modelo, necesitamos presentar algunas definiciones. Las definiciones se agrupan por tema, de manera de presentarlas en una forma más ordenada.

Ciudades

- L : conjunto de ciudades o localidades.
- LCI : conjunto de ciudades que inicialmente tienen disponible al menos un camión. $LCI \subseteq L$.
- LDI : conjunto de ciudades que inicialmente tienen disponible al menos un camionero. $LDI \subseteq L$.
- LCD_d : ciudad donde comienza el camionero d. $LDI = \bigcup_{d \in D} LCD_d$.

Camioneros

- D : conjunto de camioneros.
- DCL_k : conjunto de camioneros que comienzan su itinerario en la ciudad k. $D = \bigcup_{k \in L} DCL_k$.

Camiones

- C : conjunto de camiones.
- CCL_k : conjunto de los camiones disponibles en el momento inicial en la ciudad k. $C = \bigcup_{k \in L} CCL_k$.

Tiempo

- T : cantidad de días en el horizonte de planificación.
- K : cantidad de *momentos* en los que se divide un día. Por ejemplo, si $K = 24$, cada momento representa una hora.
- M : conjunto de momentos. Si se considera un horizonte de T días, el cardinal de M es $K \times T$ momentos, es decir $M = \{0, 1, \ldots, K \times T - 1\}$.

Pedidos

- P : conjunto de pedidos.

Rutas

- R : conjunto de *rutas*.
- $D1_{ki}$: conjunto de rutas que dejan a un camionero en la ciudad k en el momento i. Es decir, aquellas rutas en las que baja un camionero en esa ciudad y en ese momento.
- $D2_{ki}$: conjunto de rutas que dejan a dos camioneros en la ciudad k en el momento i.
- $R1_{ki}$: conjunto de rutas que requieren un camionero en la ciudad k en el momento i. Es decir, aquellas rutas en las que sube un camionero en esa ciudad y en ese momento.
- $R2_{ki}$: conjunto de rutas que requieren dos camioneros en la ciudad k en el momento i.
- RP_p : conjunto de rutas que satisfacen el pedido p.
- RCL_k : conjunto de rutas que comienzan en ciudad k.
- c_r : costo de la ruta $r \in R$.

3.2. Formulación del modelo

A continuación presentamos la formulación del modelo para el problema.

3.2.1. Variables

Las siguientes son las variables definidas para el modelo. Todas ellas son binarias.

- y_r con $r \in R$
  Indica si una ruta r es parte de la solución.
- X_{dki} con $d \in D$, $k \in L$, $i \in M$
  Indica si el camionero d se sube a un camión en la ciudad k en el momento i. En el momento inicial un camionero sólo puede subirse a un camión en la ciudad en la que se encuentra inicialmente, por lo tanto sólo existen las variables X_{dk0} si $k = LCD_d$.
- Y_{dki} con $d \in D$, $k \in L$, $i \in M$
  Indica si el camionero d descansa en la ciudad k en el momento i. Dado que en el momento inicial un camionero sólo puede descansar en la ciudad en la que se encuentra inicialmente, las variables Y_{dk0} sólo existen si $k = LCD_d$.
- W_{dki} con $d \in D$, $k \in L$, $i \in M \setminus \{0\}$
  Indica si el camionero d baja de un camión en la ciudad k en el momento i. Dado que en el momento inicial los camioneros no pueden bajarse de los camiones, estas variables sólo están definidas para $i \geq 1$.
- Z_{dj} con $d \in D$, $j \in T$
  Indica si el camionero d tiene franco el día j.

3.2.2. Función objetivo

$$\text{mín} \sum_{r \in R} c_r y_r$$

3.2.3. Restricciones

1. Todos los pedidos son satisfechos por exactamente una ruta.

$$\sum_{r \in RP_p} y_r = 1 \qquad \forall p \in P$$

2. No se utilizan más camiones de los que hay disponibles.

$$\sum_{r \in R} y_r \leq |C|$$

3. En ninguna ciudad comienzan más rutas que la cantidad de camiones que hay inicialmente en dicha ciudad.

$$\sum_{r \in RCL_k} y_r \leq |CCL_k| \qquad \forall k \in LCI$$

4. Un camionero se encuentra descansando en un momento y en una ciudad si en el momento anterior ya se encontraba descansando en esa ciudad, o si en el momento actual es dejado en esa ciudad y no sube a ningún camión.

$$Y_{dki} = Y_{dk(i-1)} - X_{dki} + W_{dki} \qquad \forall d \in D, k \in L, 1 \leq i < K \times T$$

5. En el momento inicial, los camioneros descansan en la ciudad en la que se encuentran originalmente, o se suben a un camión.

$$Y_{dk0} = 1 - X_{dk0} \qquad \forall d \in D, k = LCD_d$$

6. Los camioneros no pueden estar subiendo a un camión y descansando en el mismo momento.

$$\sum_{k \in L} (Y_{dki} + X_{dki}) \leq 1 \qquad \forall d \in D, 1 \leq i < K \times T$$

7. La cantidad de camioneros que las rutas dejan en una ciudad y en un momento se corresponde con la cantidad de camioneros que bajan de los camiones en esa ciudad y en ese momento.

$$\sum_{r \in D1_{ki}} y_r + 2 \sum_{r \in D2_{ki}} y_r = \sum_{d \in D} W_{dki} \qquad \forall k \in L, 1 \leq i < K \times T$$

8. La cantidad de camioneros que las rutas requieren en una ciudad y en un momento es igual a la cantidad de camioneros que suben a algún camión en esa ciudad y en ese momento.

$$\sum_{r \in R1_{ki}} y_r + 2 \sum_{r \in R2_{ki}} y_r = \sum_{d \in D} X_{dki} \qquad \forall k \in L, 0 \leq i < K \times T$$

9. En cualquier período de 24 horas, los camioneros descansan al menos 12 horas.

$$\sum_{k \in L} \sum_{j=i}^{i+K-1} Y_{dkj} \geq K/2 \qquad \forall d \in D, 0 \leq i < K \times T - (K-1)$$

10. En cualquier período de 7 días, los camioneros descansan al menos un día:

$$\sum_{i=j}^{j+6} Z_{di} \geq 1 \qquad \forall d \in D, 0 \leq j \leq T - 7$$

11. Si un camionero está descansando un día, descansa todos los momentos correspondientes a ese día:

$$K Z_{dj} \leq \sum_{k \in L} \sum_{i=K \times j}^{K \times j + K - 1} Y_{dki} \qquad \forall d \in D, 0 \leq j < T$$

Como se puede ver, la formulación cuenta con una gran cantidad de variables. En particular cada variable y_r se corresponde con una posible ruta del conjunto R. Cada ruta es un secuencia de eventos, llevados a cabo por un camión, que se transcurren en el tiempo desde el instante inicial hasta el horizonte de planificación. Esto hace que el número de formas de combinar estos eventos en el tiempo sea muy grande y, por ende, resulta imposible definir por extensión al conjunto de rutas. Debido a esto, consideraremos a las variables y_r en forma implícita, y trataremos modelo con el método de generación de columnas que se presenta en el capítulo 5. El resto de las variables $X_{dki}, Y_{dki}, W_{dki}$ y Z_{dj}, correspondientes al *scheduling* de camioneros, son consideradas en forma explícita en el modelo, por lo que no serán generadas.

Capítulo 4

Grafo de Rutas

El modelo de programación lineal entera que presentamos en el capítulo 3 incluye las denominadas variables de *ruta*. Cada variable y_r corresponde a una de las posibles rutas que pueden ser generadas. Recordemos que una ruta representa una secuencia de eventos que se suceden en el tiempo y que son llevados a cabo por un camión. Un evento puede corresponderse con un viaje entre ciudades, el ascenso o descenso de camioneros del camión en alguna ciudad o la realización del *pick-up* o *drop-off* de un pedido. Esta secuencia de eventos define un *schedule* para un camión.

En este capítulo presentaremos el *grafo de rutas*, una red con recursos y restricciones que permite generar secuencias de eventos realizados por un camión tales que verifican las restricciones propias del problema. Este modelo de generación nos permitirá formalizar la noción de *ruta* para el problema. Estas rutas son las que luego utilizaremos para construir las columnas asociadas a las variables y_r de la formulación del problema.

4.1. Definiciones preliminares sobre el *input* del problema

Para definir formalmente el grafo de rutas, necesitamos las siguientes definiciones relacionadas con la información del *input* del problema.

- $ciudadPkp : P \to L$
  Toma un pedido p y devuelve la ciudad en la que debe realizarse el *pick-up* de p.
- $diaPkp : P \to \{0, 1, \ldots, T-1\}$
  Toma un pedido p y devuelve el día a partir del cual se puede realizar el *pick-up* de la mercadería de p. T es la cantidad de días en el horizonte de planificación.
- $vcp : P \to \{0, 1, \ldots, K-1\}$
  Toma un pedido y devuelve el comienzo de la ventana de *pick-up*, es decir, a partir de qué momento del día se puede retirar la mercadería de p. K es la cantidad de momentos en los que se divide un día.
- $vfp : P \to \{0, 1, \ldots, K-1\}$
  Toma un pedido y devuelve el fin de la ventana de *pick-up*, es decir, hasta qué momento del día se puede retirar la mercadería de p. Asumimos que la ventana horaria de *pick-up* está comprendida en un único día, lo que implica que sólo son válidas las ventanas cuyo momento de finalización es posterior al momento de inicio: $vfp(p) > vcp(p)\ \forall p \in P$.

- $ciudadDpf : P \to L$
  Toma un pedido p y devuelve la ciudad en la que debe realizarse el *drop-off* de p.
- $diaDpf : P \to \{0, 1, \ldots, T-1\}$
  Toma un pedido p y devuelve el día a partir del cual se puede realizar el *drop-off* de la mercadería de p.
- $vcd : P \to \{0, 1, \ldots, K-1\}$
  Toma un pedido y devuelve el comienzo de la ventana de *drop-off*, es decir, a partir de qué momento del día se puede entregar la mercadería de p.
- $vfd : P \to \{0, 1, \ldots, K-1\}$
  Toma un pedido y devuelve el fin de la ventana de *drop-off*, es decir, hasta qué momento del día se puede entregar la mercadería de p. Asumimos que la ventana horaria de *drop-off* está comprendida en un único día, lo que implica que sólo son válidas las ventanas cuyo momento de finalización es posterior al momento de inicio: $vfd(p) > vcd(p)\ \forall p \in P$.
- $penalidad : P \to \mathbb{R}$
  Toma un pedido p y devuelve la penalidad asociada a realizar el *drop-off* de p en días posteriores al día especificado. Esta penalidad es por día de demora, es decir que si realizar el *drop-off* al día siguiente del esperado tiene costo c, realizarlo 3 días después tiene costo $3c$.
- $tiempo : L \times L \to \mathbb{N}_0$
  Toma dos ciudades y devuelve el tiempo que insume el viaje entre las mismas. Verifica que $tiempo(k, k) = 0\ \forall k \in L$.
- α
  Es el factor que indica el costo del viaje de un camión por unidad de tiempo.
- β
  Es el factor que indica la penalidad, por unidad de tiempo, asociada a que el camión viaje sin mercadería.

4.2. Estructura del grafo

Previamente describimos una ruta como una secuencia de eventos que define un *schedule* para un camión. Sin embargo, no cualquier secuencia de eventos va a resultar válida para el problema. Por ejemplo, una secuencia en la que un camión realiza el *drop-off* de un pedido sin antes haber realizado el correspondiente *pick-up* no es válida para nuestro problema. De aquí en adelante llamaremos *rutas* a aquellas secuencias de eventos o *schedules* que pueden ser realizados por un camión, y para generarlas utilizaremos el denominado *grafo de rutas*.

El grafo de rutas es una red dirigida con recursos y restricciones, en donde un *camino válido* desde el nodo *source* al nodo *sink* representa una posible ruta. Más adelante definiremos con precisión la noción de camino válido, pero por ahora basta con saber que es un camino dirigido que verifica las restricciones asociadas a los recursos. Los recursos que se tienen en cuenta en el grafo son dos: el tiempo y los pedidos. Un camino, a medida que va *atravesando* el grafo, acumula el tiempo de viaje del camión y los pedidos que el mismo va realizando.

A continuación se presenta una figura que brinda una noción de la estructura del grafo de rutas.

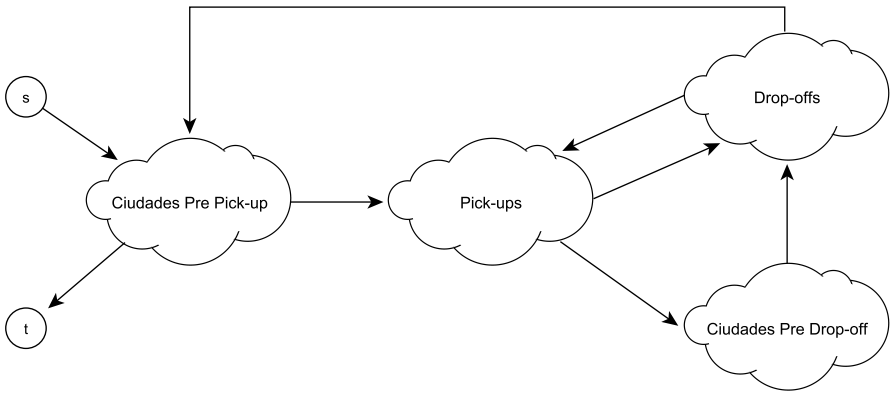

Figura 4.1: Estructura del grafo de rutas.

Los nodos distinguidos s y t representan el *source* y el *sink* del grafo de rutas. Las diferentes nubes que muestra el gráfico representan subgrafos que tienen cierta estructura en común. Es así como:

- Ciudades Pre pick-up: agrupa los nodos de las ciudades que son visitadas antes de realizar el *pick-up* de un pedido.
- Pick-ups: agrupa los nodos que representan los *pick-ups*.
- Ciudades Pre drop-off: agrupa los nodos de las ciudades que son visitadas antes de realizar el *drop-off* de un pedido.
- Drop-offs: agrupa los nodos que representan los *drop-offs*.

Esta estructura restringe los posibles recorridos que puede realizar un camión. Según el grafo de rutas, un camión comienza viajando entre las ciudades del mapa y luego procede a realizar el *pick-up* de un pedido. Una vez cargado con la mercadería puede ir directamente a realizar el *drop-off* del pedido o bien seguir viajando entre las diferentes ciudades *pre drop-off*. Después de realizar el *drop-off* de la mercadería puede ir a realizar un nuevo *pick-up* o bien comenzar nuevamente el viaje entre ciudades para repetir el proceso original.

Los ejes del grafo tienen como atributo al tiempo. Es así como un camino en el grafo de rutas irá acumulando el tiempo del viaje que realiza el camión. Se acumulará además el costo del viaje representado por el camino, y los pedidos que el camión llevará a cabo durante su recorrido.

4.2.1. Subgrafo de ciudades

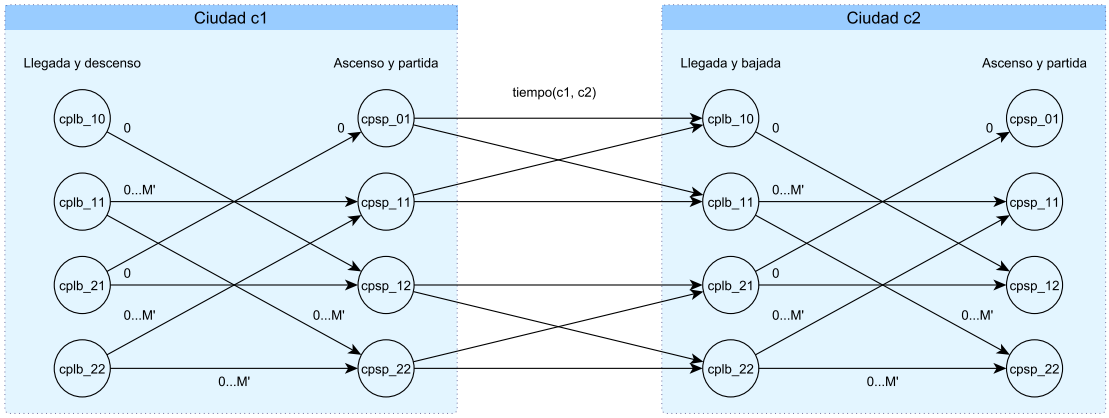

Figura 4.2: Estructura del subgrafo de ciudades.

La figura 4.2 muestra la estructura del *subgrafo de ciudades* con dos ciudades a modo de ejemplo. Cada ciudad se representa con un grafo bipartito en donde uno de los conjuntos de nodos representa la llegada del camión a la ciudad, y el otro la partida.

Cada nodo del conjunto de llegada representa una de las posibles combinaciones de cantidad de camioneros que llegan en el camión y cuántos de ellos se bajan del mismo. Cuando un camionero desciende del camión puede iniciar su descanso o ascender a otro camión que se encuentre en la ciudad. De esta forma, el nodo $cplb_{21}$ representa la situación de un camión que llega a la ciudad con dos camioneros, uno de los cuales desciende. Es importante notar que no incluimos el nodo que representa la situación en que llegan dos camioneros y ninguno desciende. Si sucediera esto, entonces el camión no podría detenerse y tampoco podrían ascender más camioneros dado que la capacidad del camión ya estaría completa. En este caso, el camión no podría realizar ninguna acción en la ciudad, por lo que su paso por allí no tiene sentido.

De la misma manera, cada nodo del conjunto de partida representa una de las posibles combinaciones de cantidad de camioneros que ascienden y cuántos de ellos parten de la ciudad en el camión. Es así como el nodo $cpsp_{12}$ representa la situación de un camión que parte de la ciudad con dos camioneros, donde uno de ellos ascendió en la partida. De la misma forma que en el conjunto de llegada, el nodo $cpsp_{02}$ no es parte del conjunto de partida, al tratarse de una situación que no puede darse bajo las condiciones impuestas por el modelo.

Cada nodo del conjunto de llegada se conecta con aquellos nodos del conjunto de

partida para los cuales se verifican las siguientes condiciones:

- Se preserva la cantidad de camioneros. Es decir que la cantidad de camioneros que llegan, menos los que descienden, sumados a los que ascienden debe ser igual a los que parten.
- No se excede el límite de dos camioneros por flota.
- Se realiza algún intercambio de flota. Es decir, se produce algún ascenso o descenso de camioneros.

Los ejes que conectan ambos conjuntos de nodos tienen el tiempo que puede transcurrir entre que el camión llega a la ciudad y parte de la misma. En la figura se muestran, sobre cada uno de estos ejes, los diferentes tiempos que puede estar detenido el camión, siendo M' el límite máximo de tiempo que un camión puede estar detenido en una ciudad. El tiempo 0 representa que el camión no se detiene entre la llegada y la partida, sino que pasa por la ciudad a realizar un intercambio de flota y sigue su camino. La notación $0 \dots M'$ expresa que un camión puede estar detenido entre 0 y M' momentos, por lo que entre los nodos correspondientes no existe uno sino múltiples ejes, cada uno correspondiente a las diferentes duraciones de la parada del camión. Entre un nodo de llegada y uno de partida puede existir el único eje con tiempo 0 o los múltiples ejes con valores de tiempo $0 \dots M'$. Esto se debe a que el modelo considera que si un camión se detiene en una ciudad todos sus camioneros deben descender del mismo. Es así como los nodos $cplb_{11}$ y $cpsp_{11}$ están conectados por los $M' + 1$ ejes posibles, ya que el camión puede quedar detenido porque no tiene camioneros a bordo. Sin embargo, los nodos $cplb_{10}$ y $cpsp_{12}$ sólo están conectados por el eje con tiempo 0, dado que en ese caso el camionero no descendió del camión.

Las ciudades en este subgrafo están conectadas por ejes que representan el viaje de un camión entre ciudades y, por ende, tienen el tiempo que toma llegar de una a otra. Estos ejes unen los nodos de partida de una ciudad con los de llegada de la siguiente, siempre que se verifique que la cantidad de camioneros que parten es igual a la cantidad de camioneros que llegan. Esto asegura que se cumpla la condición de preservación de camioneros anteriormente mencionada.

De esta manera, el subgrafo de ciudades modela el intercambio de flotas en los camiones que circulan a través del mapa. Los descensos y ascensos de camioneros en el grafo de rutas sólo se realizan en los nodos de llegada y partida de ciudad respectivamente. Los subgrafos de ciudades *pre pick-up* y *pre drop-off* son estructuralmente idénticos aunque se diferencian en el costo de sus ejes. En el caso del *pre pick-up*, el camión circula sin mercadería por lo que el costo de los ejes tiene la penalidad asociada a circular con el camión vacío. En el caso del *pre drop-off*, el camión va cargado con la mercadería que retiró al realizar el *pick-up*, por lo que los ejes no tienen costo extra.

4.2.2. Subgrafo de eventos

En la figura 4.3 se muestra la estructura del subgrafo de eventos con un pedido a modo ilustrativo. Para simplificar los términos utilizados llamaremos *evento* tanto al *pick-up* como al *drop-off* de un pedido. En el subgrafo de eventos cada pedido es un grafo bipartito donde el primer conjunto de nodos representa el comienzo del evento y el segundo, su fin.

Los nodos pertenecientes al primer conjunto representan la acción del comienzo de un *pick-up* o *drop-off* para un pedido, día y cantidad de camioneros determinados. Es así como en la figura, el nodo pc_{102} corresponde al comienzo del *pick-up* del pedido 1, realizándose en el día 0, por un camión con una flota de dos camioneros. A su vez, cada

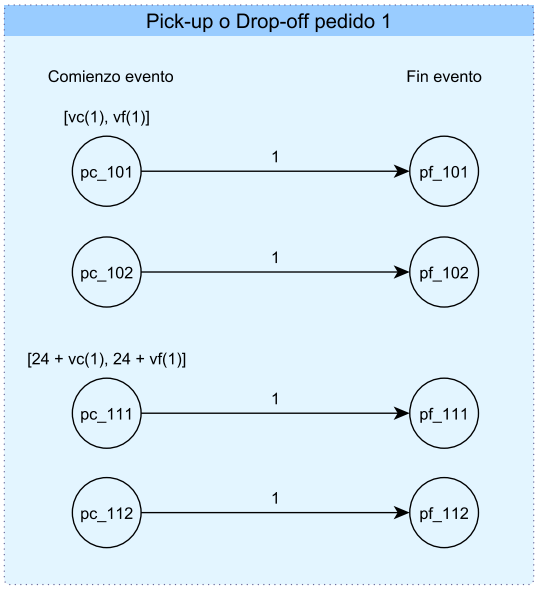

Figura 4.3: Estructura del subgrafo de los eventos de pedidos.

uno de estos nodos tiene asociada una ventana de tiempo que indica la franja horaria en la que puede llevarse a cabo el evento en cuestión. Las notaciones vc y vf representan el comienzo y fin de la ventana asociada al evento para un determinado pedido y están expresadas en valor relativo al día del evento. Los nodos de comienzo deben tener su ventana definida en términos absolutos al comienzo del horizonte de planificación, por lo que $[24 + vc(1), 24 + vf(2)]$ representa la ventana correspondiente al día 1 expresada, en este caso, en horas absolutas.

Los nodos de comienzo de evento están conectados a los nodos de fin de evento por medio de ejes que tienen el tiempo que toma la realización del evento en cuestión. Este tiempo forma parte de la definición del problema y fue fijado en 1 hora. Cada nodo de comienzo está conectado con un sólo nodo de fin correspondiente al mismo pedido, día y cantidad de camioneros que realizan el evento.

El subgrafo de eventos modela la realización de los *pick-ups* y *drop-offs* estableciendo las restricciones de ventanas de tiempo dentro de las cuales estos eventos pueden ser llevados a cabo. Los subgrafos de *pick-ups* y *drop-offs* son estructuralmente idénticos pero sus ejes tienen costos diferentes. En el caso del *pick-up*, no tienen costo alguno. En cambio, los ejes de realización de *drop-off* tienen asociado el costo de realizar el pedido fuera de fecha.

4.3. Definición del grafo

Definición 4.1 (Grafo de rutas)
El grafo de rutas es una red con recuros y restricciones definida como $GR = \langle N, E, \rho, \nu, \tau, \sigma \rangle$*, donde:*

- N *es el conjunto de nodos*
- E *es el conjunto de ejes*
- ρ *es la función asociada al recurso de los pedidos*
- ν *es la función asociada a las ventanas de tiempo tanto de pick-up como de drop-off*
- τ *es la función asociada al recurso del tiempo*
- σ *es la función asociada al costo*

A continuación presentamos en detalle los conjuntos de nodos y ejes, N y E, y definimos las funciones asociadas a los recursos y restricciones de la red, ρ, ν, τ y σ.

4.3.1. Nodos

El conjunto de nodos del grafo de rutas se define como la siguiente unión:

$$N = \{s, t\} \cup NC_{PL} \cup NC_{PP} \cup NC_{DL} \cup NC_{DP} \cup NP_C \cup NP_F \cup ND_C \cup ND_F$$

Donde s es el nodo fuente o *source*, t es el nodo sumidero o *sink*, y

Nodos llegada ciudad *pre pick-up* ($\mathbf{NC_{PL}}$)

Es el conjunto de nodos que representan la llegada a una ciudad previa a realizar un *pick-up*. Cada nodo $cplb_{klb}$ representa la situación en la que llegan l camioneros a la ciudad k, b de los cuales descenderán del camión. La cantidad de camioneros que descienden debe ser menor o igual que la cantidad de camioneros que llegan. No se define un nodo que represente la situación en la que llegan dos camioneros pero no desciende ninguno ya que en ese caso no tiene sentido que el camión pase por la ciudad. El conjunto se define como:

$$NC_{PL} = \{cplb_{k10},\ cplb_{k11},\ cplb_{k21},\ cplb_{k22} \mid k \in L\}$$

Nodos partida ciudad *pre pick-up* ($\mathbf{NC_{PP}}$)

Es el conjunto de nodos que representan la partida de una ciudad previa a realizar un *pick-up*. Cada nodo $cpsp_{ksp}$ representa la situación en la que ascienden s camioneros y la flota resultante al momento de partir de la ciudad k es de p camioneros. La cantidad de camioneros que parten de la ciudad debe ser mayor o igual que la cantidad de camioneros que ascienden al camión. No se define un nodo que represente la situación en la que no ascienden camioneros y parten dos, pues en ese caso no se debería pasar por la ciudad. El conjunto se define como:

$$NC_{PP} = \{cpsp_{k01},\ cpsp_{k11},\ cpsp_{k12},\ cpsp_{k22} \mid k \in L\}$$

Nodos llegada ciudad *pre drop-off* ($\mathbf{NC_{DL}}$)

Es el conjunto de nodos que representan la llegada a una ciudad previa a realizar un *drop-off*. Es análogo al conjunto NC_{PL}. Se define como:

$$NC_{DL} = \{cdlb_{k10},\ cdlb_{k11},\ cdlb_{k21},\ cdlb_{k22} \mid k \in L\}$$

Nodos partida ciudad *pre drop-off* ($\mathbf{NC_{DP}}$)

Es el conjunto de nodos que representan la partida de una ciudad previa a realizar un *drop-off*. Es análogo al conjunto NC_{PP}. Se define como:

$$NC_{DP} = \{cdsp_{k01},\ cdsp_{k11},\ cdsp_{k12},\ cdsp_{k22} \mid k \in L\}$$

Nodos comienzo *pick-up* ($\mathbf{NP_C}$)

Es el conjunto de nodos que representan el comienzo de un *pick-up*. Cada nodo pc_{pdc} representa la situación en la que un camión con c camioneros comienza el *pick-up* del pedido p en el día d. Se define como:

$$NP_C = \{pc_{pdc} \mid p \in P,\ diaPkp(p) \leq d < T,\ c \in \{1, 2\}\}$$

Nodos fin *pick-up* ($\mathbf{NP_F}$)

Es el conjunto de nodos que representan el fin de un *pick-up*. Es análogo al conjunto NP_C. Se define como:

$$NP_F = \{pf_{pdc} \mid p \in P,\ diaPkp(p) \leq d < T,\ c \in \{1, 2\}\}$$

Nodos comienzo *drop-off* ($\mathbf{ND_C}$)

Es el conjunto de nodos que representan el comienzo de un *drop-off*. Cada nodo dc_{pdc} representa la situación en la que un camión con c camioneros comienza el *drop-off* del pedido p en el día d. Se define como:

$$ND_C = \{dc_{pdc} \mid p \in P,\ diaDpf(p) \leq d < T,\ c \in \{1, 2\}\}$$

Nodos fin *drop-off* ($\mathbf{ND_F}$)

Es el conjunto de nodos que representan el fin de un *drop-off*. Es análogo al conjunto ND_C. Se define como:

$$ND_F = \{df_{pdc} \mid p \in P,\ diaDpf(p) \leq d < T,\ c \in \{1, 2\}\}$$

Cardinal del conjunto de nodos

El cardinal del conjunto de nodos es el siguiente:

$$|N| = 2 + 16 \times |L| + 4 \times \sum_{p \in P} T - diaPkp(p) + 4 \times \sum_{p \in P} T - diaDpf(p)$$

donde las sumatorias representan la cantidad de días que es posible realizar un *pick-up* o un *drop-off* para un determinado pedido.

4.3.2. Ejes

Se define el conjunto de ejes E del grafo de rutas como la siguiente unión:

$$E = EIC_P \cup EEC_P \cup EIC_D \cup EEC_D \cup EH_P \cup ED_P \cup EPD \cup EDP \cup ER_P \cup EH_D \cup ED_D \cup ER_D \cup ES \cup ET$$

Donde:

Ejes intra-ciudad *pre pick-up* ($\mathbf{EIC_P}$)

Es el conjunto de ejes que unen los nodos de llegada de una ciudad previa a realizar un *pick-up* con los nodos de partida de la misma. Entre cada par de nodos de llegada y partida de una ciudad ($cplb_{klb}$, $cpsp_{ksp}$) pueden existir múltiples ejes, dependiendo de la situación que represente la transición entre dichos nodos. La siguiente función define los tiempos que se corresponden con cada situación que puede darse entre los nodos de llegada y partida para cualquier ciudad $k \in L$.

Definición 4.2 (Función de tiempos ejes intra-ciudad)
La función $eic : \{0,1,2\}^4 \to \mathcal{P}(M)$ *especifica el conjunto de momentos correspondiente a cada situación de camioneros que llegan, bajan, suben y parten en una ciudad.*

$$eic(l,b,s,p) = \begin{cases} \{0\} & \text{si } l=1,\ b=0,\ s=1,\ p=2 \quad (1) \\ \{0,1,\ldots,M'\} & \text{si } l=1,\ b=1,\ s=1,\ p=1 \quad (2) \\ \{0,1,\ldots,M'\} & \text{si } l=1,\ b=1,\ s=2,\ p=2 \quad (3) \\ \{0\} & \text{si } l=2,\ b=1,\ s=0,\ p=1 \quad (4) \\ \{0\} & \text{si } l=2,\ b=1,\ s=1,\ p=2 \quad (5) \\ \{0,1,\ldots,M'\} & \text{si } l=2,\ b=2,\ s=1,\ p=1 \quad (6) \\ \{0,1,\ldots,M'\} & \text{si } l=2,\ b=2,\ s=2,\ p=2 \quad (7) \\ \emptyset & \text{en cualquier otro caso} \quad (8) \end{cases}$$

Donde $\mathcal{P}(M)$ *denota el conjunto de partes de* M*.*

(1) *Si llega un camionero y no desciende, el camión no puede detenerse*
(2) *Si el único camionero que llega desciende, el camión puede parar sin restricciones*
(3) *Si el único camionero que llega desciende, el camión puede parar sin restricciones*
(4) *Si llegan dos camioneros y sólo uno desciende, el camión no puede detenerse*
(5) *Si llegan dos camioneros y sólo uno desciende, el camión no puede detenerse*
(6) *Si descienden los dos camioneros que llegan, el camión puede parar sin restricciones*
(7) *Si descienden los dos camioneros que llegan, el camión puede parar sin restricciones*
(8) *El camión no puede pasar por una ciudad si no se presenta alguno de los casos anteriores*

Vale aclarar que si $l \neq b$*, los camioneros no descansan (i.e. la transición es instantánea) ya que para que el camión quede detenido en la ciudad todos los camioneros que llegan deben descender del mismo.*

Se define entonces el conjunto de ejes intra-ciudad *pre pick-up* de la siguiente manera:

$$EIC_P = \bigcup_{k \in L} \{(cplb_{klb},\ cpsp_{ksp})_q \mid q \in eic(l,b,s,p),\ cplb_{klb} \in NC_{PL},\ cpsp_{ksp} \in NC_{PP}\}$$

y su cardinal es:

$$|EIC_P| = (4 \times (M' + 1) + 3) \times |L| = (4 \times M' + 7) \times |L|$$

Ejes inter-ciudad *pre pick-up* ($\mathbf{EEC_P}$)

Es el conjunto de ejes que unen los nodos de partida de una ciudad previa a realizar un *pick-up* con los de llegada de otra. Sólo existe un eje entre dos de estos nodos si la cantidad de camioneros que parten de una ciudad es igual a la cantidad de camioneros que llegan a la otra. Se define como el siguiente conjunto:

$$EEC_P = \{(cpsp_{ksp},\ cplb_{mpb}) \mid cpsp_{ksp} \in NC_{PP},\ cplb_{mpb} \in NC_{PL},\ k \neq m\}$$

y su cardinal es:

$$|EEC_P| = (|L| - 1) \times 8 \times |L|$$

Ejes intra-ciudad *pre drop-off* ($\mathbf{EIC_D}$)

Es el conjunto de ejes que unen los nodos de llegada a una ciudad previa a realizar un *drop-off* con los nodos de partida de la misma ciudad. Al igual que sucede con los ejes intra-ciudad *pre pick-up*, cada par de nodos de llegada y partida de ciudad *pre drop-off* pueden estar conectados por más de un eje. Utilizando entonces la función definida en 4.2, se define el conjunto EIC_D como:

$$EIC_D = \bigcup_{k \in L} \{(cdlb_{klb},\ cdsp_{ksp})_q \mid q \in eic(l, b, s, p),\ cdlb_{klb} \in NC_{DL},\ cdsp_{ksp} \in NC_{DP}\}$$

su cardinal es:

$$|EIC_D| = (4 \times M' + 7) \times |L|$$

Ejes inter-ciudad *pre drop-off* ($\mathbf{EEC_D}$)

Es el conjunto de ejes que unen los nodos de partida de una ciudad previa a realizar un *drop-off* con los nodos de llegada de otra. Al igual que con los ejes inter-ciudad *pre pick-up*, el conjunto sólo incluye los ejes que conectan nodos en los que parten los mismos camioneros que llegan. Se define de la siguiente manera:

$$EEC_D = \{(cdsp_{ksp},\ cdlb_{mpb}) \mid cdsp_{ksp} \in NC_{DP},\ cdlb_{mpb} \in NC_{DL},\ k \neq m\}$$

y su cardinal es:

$$|EEC_D| = (|L| - 1) \times 8 \times |L|$$

Ejes hacia ciudad *pick-up* ($\mathbf{EH_P}$)

Este conjunto contiene los ejes que unen los nodos de partida de ciudades *pre pick-up* con los nodos de comienzo de *pick-up*. Sólo incluye aquellos ejes que conectan nodos en los que parten los mismos camioneros que llegan. Se define como:

$$EH_P = \{(cpsp_{ksp},\, pc_{odp}) \mid cpsp_{ksp} \in NC_{PP},\ pc_{odp} \in NP_C\}$$

y su cardinal es:

$$|EH_P| = 4 \times |L| \times \sum_{p \in P} T - diaPkp(p)$$

Ejes desde ciudad *pick-up* ($\mathbf{ED_P}$)

Este conjunto contiene los ejes que unen los nodos de fin de un *pick-up* con los nodos de llegada a una ciudad *pre drop-off*. Sólo incluye aquellos ejes que conectan nodos en los que parten los mismos camioneros que llegan. Se define como:

$$ED_P = \{(pf_{pdl}, cdlb_{klb}) \mid pf_{pdl} \in NP_F,\ cdlb_{klb} \in NC_{DL}\}$$

y su cardinal es:

$$|ED_P| = 4 \times |L| \times \sum_{p \in P} T - diaPkp(p)$$

Ejes *pick-up* a *drop-off* (EPD)

Es el conjunto de ejes que unen los fines de *pick-up* con los comienzos de *drop-off*. Representan la situación en la que un camión decide realizar la descarga de la mercadería inmediatamente después de haberla retirado, sin pasar por ninguna ciudad para hacer un intercambio en su flota. Debido a que en este caso el camión no se detiene, el *drop-off* debe realizarse obligatoriamente el mismo día en que se realizó el *pick-up* o, a lo sumo, al día siguiente. Esto es así pues la definición del problema permite un máximo de 12 hs consecutivas de trabajo. El conjunto sólo incluye los ejes que conectan nodos asociados a eventos del mismo día o días consecutivos en los que parten tantos camioneros como los que llegan.

$$EPD = \{(pf_{pdc},\, dc_{pec}) \mid pf_{pdc} \in NP_F, dc_{pec} \in ND_C,\, max(diaDpf(p), d) \leq e < min(d+2, T)\}$$

y su cardinal es:

$$|EPD| = 2 \sum_{p \in P} \sum_{d=diaPkp(p)}^{T-1} min(d+2, T) - max(diaDpf(p), d)$$

Ejes *drop-off* a *pick-up* (EDP)

Es el conjunto de ejes que unen los fines de *drop-off* con los comienzos de *pick-up*. Representan la situación en la que un camión decide realizar un nuevo retiro de mercadería

inmediatamente después de haber descargado el último pedido, sin pasar por ninguna ciudad para hacer un intercambio en su flota. Es análogo al conjunto de ejes anterior.

$$EDP = \{(df_{pdc}, pc_{qec}) \mid df_{pdc} \in ND_F,\ pc_{qec} \in NP_C,\ p \neq q,\ max(diaPkp(q), d) \leq e < min(d+2, T)\}$$

y su cardinal es:

$$|EDP| = 2 \sum_{p \in P} \sum_{d=diaPkp(p)}^{T-1} \sum_{q \in P,\ q \neq p} min(d+2, T) - max(diaDpf(q), d)$$

Ejes realización de *pick-up* ($\mathbf{ER_P}$)

Es el conjunto de ejes que unen los nodos de comienzo de *pick-up* con los correspondientes nodos de fin de *pick-up*. Se define de la siguiente manera:

$$ER_P = \{(pc_{pdc}, pf_{pdc}) \mid pc_{pdc} \in NP_C,\ pf_{pdc} \in NP_F\}$$

su cardinal es:

$$|ER_P| = 2 \times \sum_{p \in P} T - diaPkp(p)$$

Ejes hacia ciudad *drop-off* ($\mathbf{EH_D}$)

Este conjunto contiene los ejes que unen los nodos de partida de ciudades *predrop-off* con los nodos de comienzo de *drop-off*. Es análogo al conjunto de ejes hacia ciudad *pick-up*. Se define como:

$$EH_D = \{(cdsp_{ksp}, dc_{odp}) \mid cdsp_{ksp} \in NC_{DP},\ dc_{odp} \in ND_C\}$$

y su cardinal es:

$$|EH_D| = 4 \times |L| \times \sum_{p \in P} T - diaDpf(p)$$

Ejes desde ciudad *drop-off* ($\mathbf{ED_D}$)

Este conjunto contiene los ejes que unen los nodos de fin de un *drop-off* con los nodos de llegada a una ciudad *pre pick-up*. Es análogo al conjunto de ejes desde ciudad *pick-up*. Se define como:

$$ED_D = \{(df_{pdl}, cplb_{klb}) \mid df_{pdl} \in ND_F,\ cplb_{klb} \in NC_{PL}\}$$

y su cardinal es:

$$|ED_D| = 4 \times |L| \times \sum_{p \in P} T - diaDpf(p)$$

Ejes realización de *drop-off* ($\mathbf{ER_D}$)

Es el conjunto de ejes que unen los nodos de comienzo de *drop-off* con los correspondientes nodos de fin de *drop-off*. Se define de la siguiente manera:

$$ER_D = \{(dc_{pdc}, df_{pdc}) \mid dc_{pdc} \in ND_C,\ df_{pdc} \in ND_F\}$$

su cardinal es:

$$|ER_D| = 2 \times \sum_{p \in P} T - diaDpf(p)$$

Ejes *source* (ES)

Este conjunto contiene los ejes que parten del nodo *source*. Conectan el nodo s con los nodos de partida de ciudades *pre pick-up* que sean ubicación inicial de algún camión y en las que asciendan los mismos camioneros que parten. Entre el nodo *source* y cada nodo de partida de ciudad existen tantos ejes como momentos en el horizonte de planificación. Cada uno de estos ejes tiene la duración de un momento específico en el horizonte, lo que permite que una ruta comience en cualquier instante de tiempo. Recordemos que LCI representa el conjunto de ciudades en las cuales hay algún camión en el momento inicial.

$$ES = \{(s, cpsp_{kcc})_q \mid k \in LCI,\ cpsp_{kcc} \in NC_{PP},\ q \in M\}$$

y su cardinal es:

$$|ES| = 2 \times |LCI| \times K \times T$$

Ejes *sink* (ET)

Este conjunto contiene los ejes que llegan al nodo *sink*. Sólo se puede alcanzar el nodo t desde la llegada a una ciudad *pre pick-up* en donde desciendan todos los camioneros que llegan. Esto es así pues para que el camión termine su recorrido, es necesario que desciendan todos sus camioneros. Se define de la siguiente manera:

$$ET = \{(cplb_{kll}, t) \mid cplb_{kll} \in NC_{PL}\}$$

y su cardinal es:

$$|ET| = 2 \times |L|$$

Cardinal del conjunto de ejes

El cardinal del conjunto de ejes es el siguiente:

$$\begin{aligned}|E| = \quad & 2 \times |L| \times (4 \times M' + 8 \times |L|) \\ & +(2 \times \textstyle\sum_{p \in P} T - diaPkp(p)) \times (4 \times |L| + 1) \\ & +(2 \times \textstyle\sum_{p \in P} T - diaDpf(p)) \times (4 \times |L| + 1) \\ & +2 \textstyle\sum_{p \in P} \sum_{d=diaPkp(p)}^{T-1} min(d+2, T) - max(diaDpf(p), d) \\ & +2 \textstyle\sum_{p \in P} \sum_{d=diaDpf(p)}^{T-1} \sum_{q \in P,\, q \neq p} min(d+2, T) - max(diaDpf(q), d) \\ & +2 \times |LCI| \times K \times T\end{aligned}$$

4.3.3. Funciones asociadas a recursos y restricciones

A continuación se definen las funciones asociadas a los recursos y restricciones del grafo de rutas. Estas funciones nos permirán definir la noción de *validez* de un camino.

Definición 4.3 (Función *pick-up* realizado)
La función pick-up realizado *se define sobre los nodos de fin de pick-up y devuelve el pedido correspondiente a dicho pick-up.*

$$\begin{aligned}\rho &: NP_F \to P \\ \rho(pf_{pdc}) &= p\end{aligned}$$

Definición 4.4 (Función ventana de tiempo para realización de evento)
La función ventana de tiempo para realización de evento *se define sobre los nodos de comienzo de pick-up y drop-off. Devuelve el intervalo de tiempo en que se puede visitar un determinado nodo de comienzo de evento.*

$$\nu : NP_c \cup ND_c \to \{[t_1, t_2] \mid t_1, t_2 \in M, 0 \leq t_1 < t_2 < K \times T\}$$

$$\nu(ec_{pdc}) = \begin{cases} [d \times K + vcp(p),\ d \times K + vfp(p)] & \text{si } ec_{pdc} \in NP_C \\ [d \times K + vcd(p),\ d \times K + vfd(p)] & \text{si } ec_{pdc} \in ND_C \end{cases}$$

vcp representa el momento del comienzo de una ventana relativo a un día, por lo cual es necesario sumarle $d \times N$ para calcular el momento en el cual comienza la ventana de un día d en términos absolutos.

Recordemos que K representa la cantidad de momentos en los que se divide un día, y que la carga y descarga de mercadería toma una hora, es decir, $K/24$ momentos.

Definición 4.5 (Función tiempo)
La función tiempo *se define sobre el conjunto de ejes E y devuelve el tiempo que toma*

transitar un determinado eje del grafo de rutas.

$$\tau : E \to \mathbb{N}_0$$

$$\tau(e) = \begin{cases} q & \text{si } e = (cel_{klb}, cep_{ksp})_q \in EIC_P \cup EIC_D \\ tiempo(k,\, m) & \text{si } e = (cep_{ksp}, cel_{mpb}) \in EEC_P \cup EEC_D \\ tiempo(k,\, ciudadPkp(p)) & \text{si } e = (cpsp_{ksp}, pc_{odp}) \in EH_P \\ tiempo(ciudadPkp(p),\, k) & \text{si } e = (pf_{pdc}, cdlb_{klb}) \in ED_P \\ K/24 & \text{si } e = (ec_{pdc}, ef_{pdc}) \in ER_P \cup ER_D \\ tiempo(k,\, ciudadDpf(p)) & \text{si } e = (cdsp_{ksp}, dc_{odp}) \in EH_D \\ tiempo(ciudadDpf(p),\, k) & \text{si } e = (df_{odc}, cplb_{ksp}) \in ED_D \\ tiempo(ciudadPkp(p),\, ciudadDpf(p)) & \text{si } e = (pf_{pdc}, dc_{pec}) \in EPD \\ tiempo(ciudadDpf(p),\, ciudadPkp(o)) & \text{si } e = (df_{pdc}, pc_{oec}) \in EDP \\ q & \text{si } e = (s, cpsp_{kcc})_q \in ES \\ 0 & \text{si } e = (cplb_{kll}, t) \in ET \end{cases}$$

Donde q, como subíndice de un eje, representa el tiempo que toma la transición entre aquellos nodos que pueden tener múltiples ejes entre sí (ver definición 4.2).

Definición 4.6 (Función costo)
La función costo *se define sobre el conjunto de ejes E y devuelve el costo que insume transitar un determinado eje del grafo de rutas.*

$$\sigma : E \to \mathbb{N}_0$$

$$\sigma(e) = \begin{cases} (d - diaDpf(p)) \times penalidad(p) & si\ e = (dc_{pdc}, df_{pdc}) \in ER_D \\ (\alpha + \beta) \times tiempo(k,\, m) & si\ e = (cpsp_{ksp}, cplb_{mpb}) \in EEC_P \\ (\alpha + \beta) \times tiempo(k,\, ciudadPkp(p)) & si\ e = (cpsp_{ksp}, pc_{odp}) \in EH_P \\ (\alpha + \beta) \times tiempo(ciudadDpf(p),\, ciudadPkp(q)) & si\ e = (df_{pdc}, pc_{qec}) \in EDP \\ (\alpha + \beta) \times tiempo(ciudadDpf(p),\, k) & si\ e = (df_{pdl}, cplb_{klb}) \in ED_D \\ \beta \times tiempo(k, m) & si\ e = (cdsp_{ksp},\, cdlb_{mpb}) \in EEC_D \\ \beta \times tiempo(ciudadPkp(p),\, k) & si\ e = (pf_{pdl}, cdlb_{klb}) \in ED_P \\ \beta \times tiempo(ciudadPkp(p),\, ciudadDpf(p)) & si\ e = (pf_{pdc}, dc_{pec}) \in EPD \\ \beta \times tiempo(k,\, ciudadDpf(p)) & si\ e = (cdsp_{ksp}, dc_{odp}) \in EH_D \\ 0 & en\ otro\ caso \end{cases}$$

Donde α indica el costo de que un camión circule sin mercadería y β indica el costo de viaje por unidad de tiempo.

Los ejes pueden tener tres tipos de costos: el costo del viaje, la penalidad asociada a realizar un *drop-off* en algún día posterior al primer día habilitado, y la penalidad correspondiente a viajar con el camión sin mercadería. El primero está incluido sobre todos los ejes que representan un viaje entre dos ciudades. El segundo está incluido en aquellos ejes que representan la realización de un *drop-off*. El último tipo de costo está incluido en los ejes que representen viajes entre ciudades donde el camión no lleva mercadería.

4.4. Rutas

Una vez definido formalmente el grafo de rutas GR estamos en condiciones de definir el concepto de *ruta*. Para esto utilizaremos el concepto de camino válido en GR. Un camino en GR se representa como una secuencia de ejes de la siguiente manera:

$$C = [(n_1, n_2), (n_2, n_3), \ldots, (n_{m-1}, n_m)]$$

Para formalizar la idea de camino válido precisamos definir las funciones de *tiempo acumulado* de un camino y *pick-ups realizados* por un camino.

Definición 4.7 (Función tiempo acumulado)
La función de tiempo acumulado *toma un camino y devuelve el tiempo que acumula el mismo. Se define recursivamente como:*

$$T([(s,n)]) = \tau((s,n))$$

$$T(C \oplus (n,ec)) = \begin{cases} max(T(C) + \tau((n,ec)), t_1) & \text{si } (n,ec) \in EH_P \cup EH_D \cup EPD \cup EDP, \\ & \text{siendo } [t_1,t_2] = \nu(ec) \\ T(C) + \tau((n,ec)) & \text{si no} \end{cases}$$

Donde $\oplus$ representa el operador que agrega un eje al final de un camino. Notar que el caso base está definido en base al camino con un solo eje perteneciente al conjunto ES dado que todos los caminos en GR deben comenzar en el nodo source.

Es importante notar que la función de tiempo acumulado de un camino no se define simplemente como la suma de los tiempos de sus ejes. Esto es así pues es necesario tener en cuenta los tiempos de espera que pueden producirse en los nodos que tienen una ventana de tiempo (nodos de comienzo de evento).

Definición 4.8 (Función pick-ups realizados)
La función de pick-ups realizados toma un camino y devuelve la secuencia de pedidos cuyos pick-ups fueron realizados por el camino. Se define recursivamente como:

$$P([(s,n)]) = [\,]$$

$$P(C \oplus (ec,ef)) = \begin{cases} P(C) \oplus p & \text{si } (ec,ef) = (pc_{pdc}, pf_{pdc}) \in ER_P \\ P(C) & \text{si no} \end{cases}$$

4.4.1. Camino válido

Representaremos el concepto de *ruta* como un *camino válido* en el grafo de rutas. Un camino C es válido si comienza en el nodo s, finaliza en el nodo t y cumple las siguientes propiedades:

- **Respeta todas las ventanas de tiempo de sus nodos**
  Para cada nodo de comienzo de *pick-up* o de *drop-off* en el camino, se debe verificar que el tiempo acumulado desde el comienzo del camino hasta dicho nodo no supere la cota superior de la ventana de tiempo del nodo. Esto impide que se realice un evento fuera de la ventana de tiempo correspondiente.

$$\forall\ i \text{ tal que } 1 \leq i \leq |C|,$$
$$\text{si } C[i] = (n, ec) \in EH_P \cup EH_D \cup EPD \cup EDP$$
$$\Rightarrow$$
$$t_1 \leq T(C[1,\ldots,i]) \leq t_2, \text{ siendo } \nu(ec) = [t_1, t_2]$$

- **Respeta la secuencia de *pick-ups* y *drop-offs***
  Para cada nodo de comienzo de *drop-off*, el último *pick-up* realizado corresponde al mismo pedido y no a otro.

$$\forall\ i \text{ tal que } 1 \leq i \leq |C|,$$
$$\text{si } C[i] = (n, dc_{pdc}) \in EH_D \cup EPD$$
$$\Rightarrow$$
$$p = ultimo(P(C[1,\ldots,i]))$$

Donde *ultimo* devuelve el último pedido en la secuencia de pedidos realizados por el camino.

- **No realiza el *pick-up* de un pedido que ya realizó previamente**
  Para cada nodo de comienzo de *pick-up* se debe verificar que el pedido correspondiente al nodo no haya sido realizado en el camino.

$$\begin{gathered}\forall\ i \text{ tal que } 1 \leq i \leq |C|, \\ \text{si } C[i] = (n, pc_{pdc}) \in EH_P \cup EDP \\ \Rightarrow \\ p \notin P(C[1, \ldots, i]))\end{gathered}$$

- **El tiempo acumulado por el camino no supera el horizonte de tiempo**

$$T(C) < K \times T$$

Definición 4.9 (Costo de un camino válido)
Se define la función de costo de un camino válido *como la suma de los costos de los ejes que lo conforman.*

$$\Sigma(C) = \sum_{e \in C} \sigma(e)$$

4.4.2. Información de una ruta

Una ruta debe brindar la información de los pedidos que realiza y de cuándo y en qué ciudad se detiene para que asciendan y desciendan camioneros. A continuación definimos algunas funciones que nos permiten obtener esta información de una ruta.

- $pedidos : R \to \mathcal{P}(P)$
  Toma una ruta r y devuelve el conjunto de pedidos que son satisfechos por r.
- $momPkp : R \times P \to M$
  Toma una ruta r y un pedido p realizado por r y devuelve el momento en que se lleva a cabo el *pick-up* de p.
- $momDpf : R \times P \to M$
  Toma una ruta r y un pedido p realizado por r y devuelve el momento en que se lleva a cabo el *drop-off* de p.
- $camAsc : R \times L \times M \to \{0, 1, 2\}$
  Toma una ruta r, una ciudad c y un momento m, y devuelve la cantidad de camioneros que ascienden al camión en c en el instante m.
- $camDesc : R \times L \times M \to \{0, 1, 2\}$
  Toma una ruta r, una ciudad c y un momento m, y devuelve la cantidad de camioneros que descienden del camión en c en el instante m.
- $c_r : R \to \mathbb{R}$
  Toma una ruta r y devuelve su costo. Este costo corresponde al costo Σ del camino válido que representa la ruta.

Capítulo 5

Generación de columnas

En el capítulo 3 introdujimos la formulación del modelo de programación lineal entera para resolver el VCSPDPTW. Dicha formulación incluye una variable y_r por cada posible ruta incluida en el conjunto R. Recordemos que una ruta representa una secuencia de eventos llevados a cabo por un camión en el transcurso del tiempo y contiene la información del momento en el que se produce cada evento. Como consecuencia, la cantidad de rutas posibles es muy grande, lo cual en términos prácticos hace imposible generar la matriz A completa.

Por esto decidimos utilizar la técnica de generación de columnas para generar las columnas asociadas a las variables y_r. Para generar cada una de estas columnas es necesario construir previamente la ruta que la variable representa. El grafo de rutas presentado en el capítulo anterior nos brinda la estructura para generar rutas, pero precisamos de un algoritmo que nos permita obtener rutas tales que sus columnas asociadas, de ser incluidas en la base, contribuyan a disminuir el valor de la función objetivo de la relajación del problema maestro. Es importante aclarar que el proceso de generación de columnas se realiza únicamente para la resolución de la relajación del nodo raíz del esquema *Branch and Cut*. En las siguientes secciones presentaremos la descripción del algoritmo que utilizamos para generar las rutas y cómo la información de estas rutas se transformará en columnas de la matriz de restricciones A.

5.1. Generación de columnas

La matriz de restricciones A está compuesta por columnas asociadas a las variables de la formulación del modelo: X_{dki}, Y_{dki}, W_{dki}, Z_{dj} y y_r. Son estas últimas variables binarias las que indican qué rutas están incluidas en la solución y, por ende, será necesario generar las columnas A_{y_r} asociadas a dichas variables.

Una columna A_{y_r} es un vector que contiene ceros, unos o dos según el coeficiente que acompañe a la variable y_r en la correspondiente restricción. El vector columna A_{y_r}, correspondiente a la ruta y_r, tiene la siguiente forma:

$$A^t_{y_r} = (\underbrace{[0,1],...,[0,1]}_{1},\underbrace{1}_{2},\underbrace{[0,1],...,[0,1]}_{3},\underbrace{0,...,0}_{4},\underbrace{0,...,0}_{5},\underbrace{0,...,0}_{6},\underbrace{[0,1,2],...,[0,1,2]}_{7},$$
$$\underbrace{[0,1,2],...,[0,1,2]}_{8},\underbrace{0,...,0}_{9},\underbrace{0,...,0}_{10},\underbrace{0,...,0}_{11})$$

donde:

1. $|P|$ filas, donde la fila i indica si la ruta realiza (1) o no (0) el pedido número i. Se corresponden con las restricciones tipo 1 del modelo.
2. Indica que la ruta es llevada a cabo por un camión. Siempre está en 1 y se corresponde con la restricción tipo 2 del modelo.
3. $|LCI|$ filas que indican en qué ciudad, de las que tienen al menos un camión en el instante inicial, comienza la ruta. Se corresponden con restricciones de tipo 3 del modelo.
4. $|D| \times |L| \times K \times (T-1)$ filas que indican la relación entre ascensos y descensos de camioneros, por lo que las rutas no influyen en estas filas. Se corresponden a las restricciones de tipo 4 del modelo.
5. $|D|$ filas que establecen las condiciones iniciales sobre los camioneros por lo que las rutas no influyen en estas filas. Se corresponden con las restricciones de tipo 5 del modelo.
6. $|D| \times K \times (T-1)$ filas que establecen que los camioneros no pueden estar descansando y trabajando al mismo tiempo. Tampoco son afectadas por las rutas. Se corresponden con las restricciones de tipo 6 del modelo.
7. $|L| \times K \times (T-1)$ filas que contienen un 0, 1 ó 2 según cuántos camioneros descienden de un camión (son dejados por la ruta) en una ciudad y un momento determinados. Se corresponden con las restricciones de tipo 7 en el modelo.
8. $|L| \times K \times T$ filas que contienen un 0, 1 ó 2 según cuántos camioneros ascienden a un camión (son requeridos por la ruta) en una ciudad y momento determinado. Se corresponden con las restricciones de tipo 8 del modelo.

9, 10 y 11. Estas filas corresponden a las restricciones de tipo 9, 10 y 11 del modelo y establecen las condiciones de descanso de los camioneros. No son afectadas por las rutas.

La columna que representa a la ruta debe ser generada de tal manera de garantizar que su inclusión en la base mejore la función objetivo. Siguiendo la notación del método *Simplex Revisado* y dado que se trata de un problema de minimización, se busca generar una ruta A_{y_r} que verifique lo siguiente:

$$c_r - y^* A_{y_r} < 0 \tag{5.1}$$

donde c_r es el costo de la ruta r e y^* es el vector con los valores de las variables duales (asociadas a las restricciones del problema) de la última iteración del método *Simplex*.

Como mencionamos en el capítulo anterior, un camino válido en el grafo de rutas representa una ruta. Para poder generar rutas candidatas a ingresar a la base es necesario que la eficiencia de un camino válido pueda medirse en términos de la información que nos brinda *Simplex* luego de resolver cada relajación. Si el costo de un camino válido en el grafo de rutas resultara ser exactamente $c_r - y^* A_{y_r}$ y dicho costo fuera negativo, la ruta correspondiente al camino verificaría la condición (5.1) necesaria para ingresar a la base. De esta manera el problema de generar una ruta que sea candidata a ingresar a la base se transforma en el problema de encontrar caminos válidos de costo negativo en el grafo de rutas.

Una forma de encarar la resolución de este problema consiste en buscar el camino de costo mínimo en el grafo de rutas. Si este costo resultara ser positivo, sabríamos que no existe ninguna ruta que, de ser incluída en el problema, mejoraría el valor de la función objetivo (i.e. no existe columna A_{y_r} tal que verifica la condición 5.1). Si en cambio, el

costo del camino fuese negativo, tendríamos la garantía de que la columna resultante es candidata a ingresar a la base en la siguiente relajación.

El *Shortest Path Problem* (SPP) o problema de camino mínimo en un grafo puede resolverse en tiempo polinomial utilizando algoritmos conocidos. Sin embargo, estos algoritmos no son aplicables al problema de encontrar un camino de costo mínimo en el grafo de rutas. En el grafo de rutas, un camino válido debe cumplir ciertas propiedades asociadas a los recursos del grafo. Esto transforma al SPP sobre el grafo de rutas en el problema de camino mínimo con restricciones sobre recursos (SPPRC). Para resolverlo desarrollamos un algoritmo similar al propuesto por Dijkstra para el SPP pero que considera las propiedades que deben cumplir los caminos en relación a los recursos.

5.1.1. Introducción de los valores de las variables duales en el grafo de rutas

Para que el costo de un camino válido en el grafo de rutas sea $c_r - y^* A_{y_r}$ debemos introducir en el grafo los valores de las variables duales correspondientes a la relajación resuelta mediante *Simplex*.

El costo de un camino válido está dado por la suma del costo de los ejes que lo componen. A su vez, cada eje tiene un costo diferente dependiendo de la *acción* que el mismo represente. Para definir el costo *real* de un eje fue necesario introducir el concepto de costo dual fijo y costo dual variable. El primero hace referencia a los costos de acciones que no varían según el tiempo en que se realizan: comenzar una ruta en una ciudad determinada o realizar un pedido. El segundo hace referencia a los costos que varían según el tiempo en que sus acciones asociadas se realizan: los ascensos y descensos de los camioneros.

Se definen entonces las siguientes funciones donde n representa el número de restricciones del problema, L el conjunto de ciudades, P el conjunto de pedidos, LCI el conjunto de ciudades con al menos un camión en el instante inicial y M el conjunto de momentos.

- $cdfComRuta : \mathbb{R}^n \times LCI \to \mathbb{R}$.
  Toma el vector de valores duales y^* junto con una ciudad k. Devuelve el costo dual fijo asociado a que una ruta comience en la ciudad k.
- $cdfPedido : \mathbb{R}^n \times P \to \mathbb{R}$.
  Toma el vector de valores duales y^* junto con un pedido p. Devuelve el costo dual fijo asociado a que una ruta realice el pedido p.
- $cdvAscCam : \mathbb{R}^n \times L \times M \to \mathbb{R}$.
  Toma el vector de valores duales y^*, una ciudad k y un momento m. Devuelve el costo dual variable asociado a que asciendan camioneros en la ciudad k y en el momento m.
- $cdvDescCam : \mathbb{R}^n \times L \times M \to \mathbb{R}$.
  Toma el vector de valores duales y^*, una ciudad k y un momento m. Devuelve el costo dual variable asociado a que desciendan camioneros en la ciudad k y en el momento m.

Con los conceptos de costos duales fijos y variables ya introducidos, podemos definir lo que denominaremos el costo *real* de un eje.

Definición 5.1 (Función costo real)
La función de costo real *se define sobre el conjunto de ejes E del grafo de rutas, el conjunto*

de momentos M y el vector de valores duales. Se define como la siguiente función partida:

$$\sigma^* : E \times M \times \mathbb{R}^n \to \mathbb{R}$$

$$\sigma^*(e,m,y) = \begin{cases} \sigma(e) - (cdfComRuta(y,k) + \\ \quad c \times cdvAscCam(y,k,m)) & \text{si } e = (s, cpp_{kcc}) \in ES \\ \sigma(e) - cdfPedido(y,p) & \text{si } e = (dc_{pdc}, df_{pdc}) \in ER_D \\ \sigma(e) - s \times cdvAscCam(y,k,m+\tau(e)) & \text{si } e = (cl_{klb}, cp_{ksp}) \in EIC_P \cup EIC_D \\ \sigma(e) - b \times cdvDescCam(y,h,m+\tau(e)) & \text{si } e = (cp_{ksp}, cl_{hlb}) \in EEC_P \cup EEC_D \\ \sigma(e) - b \times cdvDescCam(y,k,m+\tau(e)) & \text{si } e = (f_{pdl}, cl_{klb}) \in ED_P \cup ED_D \\ \sigma(e) & \text{en otro caso} \end{cases}$$

Donde:

- *e es un eje del grafo de rutas*
- *m es el momento en que se comienza a transitar el eje*
- *y es el vector de valores duales*
- *σ es la función original de costo para un eje*
- *τ es la función de tiempo para un eje*

El primer caso trata sobre los ejes que salen del nodo *source*. Estos ejes representan los comienzos de las rutas en las diferentes ciudades, por lo que su costo va a estar dado por la resta entre el costo original del eje y los costos duales asociados a comenzar la ruta en la correspondiente ciudad y a los camioneros que ascienden al camión.

El segundo caso trata sobre los ejes de realización de *drop-off*. Estos ejes representan la realización de un pedido por parte de la ruta y su costo va a depender del costo dual fijo de realizar el correspondiente pedido.

El tercer caso contempla las partidas de las ciudades, y los ejes que representan esta acción en el grafo son los de intra-ciudad. Su costo está dado por la resta entre el costo original del eje y el costo dual variable del ascenso de camioneros en la partida de la ciudad. Es importante notar que el ascenso de camioneros se produce en el fin de la transición del eje, o sea, en el nodo de partida de ciudad. Es por esto que se toma el momento de la partida (momento en que se comienza a transitar el eje sumado al tiempo que toma la transición) para determinar el costo del ascenso.

Los casos cuatro y cinco son análogos al tercero excepto que tratan sobre las llegadas a las ciudades, tanto desde nodos de partida de ciudad como de fin de evento. Es por esto que dependen del costo dual variable de descenso de camioneros.

Definición 5.2 (Función costo real acumulado)
Dado un camino C en el grafo de rutas, se define la función de costo real acumulado *como la suma de los costos reales de los ejes que lo conforman.*

$$\Sigma^*(C) = \sum_{e \in C} \sigma^*(e)$$

5.1.2. Algoritmo para el RCSPP

El algoritmo que desarrollamos está basado en el presentado por Faramroze, Nemhauser y Savelsbergh en [ENS08]. Se trata de un algoritmo basado en la técnica de programación

dinámica que resuelve el RCSPP en forma exacta, es decir, que obtiene el camino válido de costo mínimo en el grafo de rutas.

Para comprender la descripción del algoritmo es preciso que establezcamos primero cuándo consideramos que un camino es *mejor* que otro. La noción de validez de un camino en el grafo de rutas ya fue presentada en la sección 4.4.1 del capítulo anterior, por lo que asumiremos que los caminos con los que tratamos ahora son válidos según esta definición.

La dominancia es una relación definida entre caminos y sirve para determinar si un camino es mejor que otro. A diferencia de un algoritmo de camino mínimo común y ordinario, en el cual la relación de dominancia está dada por el costo de los caminos, en un grafo con recursos y restricciones, como es el grafo de rutas, la relación de dominancia resulta más compleja.

Definición 5.3 (Relación de dominancia)
Sean C_1 y C_2 dos caminos en el grafo de rutas GR, C_1 domina a C_2 ($C_1 \preccurlyeq C_2$) sii se cumplen las siguientes condiciones:

- *C_1 y C_2 finalizan en el mismo nodo*
- *El costo real de C_1 es menor o igual al de C_2 : $\Sigma^*(C_1) \leq \Sigma^*(C_2)$*
- *El tiempo de C_1 es igual al de C_2 : $T(C_1) = T(C_2)$*
- *Si ambos caminos tienen pendiente el drop-off del último pedido:*
  - ◇ *$ultimo(P(C_1)) = ultimo(P(C_2))$*

  Si no
  - ◇ *Tanto C_1 como C_2 no deben tener pendiente el drop-off del último pedido realizado*
- *Los pedidos realizados por C_1 están incluidos en los de C_2 : $P(C_1) \subseteq P(C_2)$*

Estas condiciones garantizan que cada extensión válida de un camino C_2 también resulte una extensión válida de C_1, por lo que cada secuencia de acciones que pueda continuar realizando el camino C_2 también podrá ser realizada por C_1. Además, el camino C_1 tiene un costo menor al de C_2, por lo que se puede afirmar que C_1 es un mejor camino que C_2 en cuanto a su costo real acumulado.

La relación de dominancia sirve para que el algoritmo descarte un camino C_2 si es que existe otro camino C_1 tal que $C_1 \preccurlyeq C_2$. Por supuesto que esta relación no es de orden total, por lo que gran parte de los caminos del grafo de rutas no van a dominar ni ser dominados por otros y, en consecuencia, no podrán ser descartados. En esto radica la complejidad del problema de buscar un camino mínimo dominante de mejor costo: en todo momento la cantidad de caminos no dominados por ningún otro puede ser muy grande. Esto se traduce en que el algoritmo debe analizar un número muy grande, posiblemente exponencial, de extensiones de cada uno de ellos.

Pseudocódigo

A continuación se presenta el pseudocódigo del algoritmo para resolver el RCSPP sobre el grafo de rutas. Recordemos que E es el conjunto de ejes del grafo de rutas y ES el conjunto de ejes que parten del nodo *source*.

Algoritmo 4 RCSPP para el grafo de rutas

Input: GR grafo de rutas
Output: C^* el camino mínimo en GR

1: *// Inicializar el conjunto de caminos a procesar con los ejes salientes del source*
2: $CAP \leftarrow ES$
3: *// Inicializar el conjunto de caminos no dominados como vacío*
4: $CND \leftarrow \emptyset$
5: **while** $CAP \neq \emptyset$ **do**
6: Elegir camino $C \in CAP$ y eliminarlo de CAP
7: **for all** eje $e \in E$ que extienda en forma factible a C **do**
8: **if** $\nexists\, C' \in CAP \cup CND$ tal que $C' \preccurlyeq [C, e]$ **then**
9: *// Si $[C, e]$ no es dominado, eliminar los caminos a los que él domina*
10: Eliminar $C' \in CAP \cup CND$ tal que $[C, e] \preccurlyeq C'$
11: *// Agregar el camino extendido a los caminos a procesar*
12: $CAP \leftarrow CAP \cup \{[C, e]\}$
13: **end if**
14: **end for**
15: *// Agregar el camino procesado a los caminos no dominados*
16: $CND \leftarrow CND \cup \{C\}$
17: **end while**
18: **return** $C^* = \arg\min \{costo(C) : C \in CND$ y finaliza en el *sink*$\}$

El algoritmo comienza inicializando un conjunto de caminos a procesar, que contiene los caminos triviales conformados por los ejes que salen del nodo *source*, y un conjunto vacío de caminos no dominados. Luego selecciona un camino C del conjunto de caminos a procesar y lo extiende con cada eje saliente del último nodo de C. Es importante aclarar que aquí sólo se tienen en cuenta aquellas extensiones *factibles*, es decir, las extensiones que conservan la propiedad de validez del camino. Si una extensión de C no resulta factible, es descartada en este momento.

Para cada extensión factible de C, el algoritmo verifica si en el conjunto de caminos no dominados existe algún camino que la domine. Si es así, ya se cuenta con un mejor camino que finaliza en el mismo nodo que dicha extensión, por lo que no vale la pena seguir explorándola. En caso contrario, elimina todos aquellos caminos que son dominados por la extensión de C y la agrega al conjunto de caminos a procesar para que continúe extendiéndose.

De esta manera, el algoritmo va construyendo extensiones factibles de caminos que parten del nodo *source*, descartando aquellos caminos que resultan inútiles. El ciclo principal se repite hasta que ya no existan caminos que puedan ser procesados. El algoritmo devuelve el camino no dominado con costo mínimo entre aquellos que finalizan en el nodo *sink*.

5.1.3. Variantes para la generación de rutas

Si bien el algoritmo propuesto genera, en caso de que exista, un camino de costo negativo, en la práctica presenta dos tipos de problemas. Por un lado, en muchos casos el algoritmo demora mucho tiempo en terminar, y por otro lado, muchas veces la columna asociada a la ruta generada no forma parte de una solución factible entera.

El principal motivo por el cual el algoritmo demora mucho en obtener el resultado es la gran cantidad de caminos que deben ser analizados. Esto se debe principalmente a

dos factores: por un lado, el grafo de rutas generado a partir del mapa de ciudades y el conjunto de pedidos cuenta con una gran cantidad de nodos y ejes. De hecho, para un problema de 3 ciudades, 3 pedidos y un horizonte de planificación de 2 días, el tamaño del grafo es de casi 100 nodos y más de 1000 ejes. En consecuencia, existen muchos caminos válidos entre el nodo *source* y el *sink* que deberían ser considerados por el algoritmo.

Por otro lado, a pesar de que el algoritmo descarta aquellos caminos que son dominados, son muchos los caminos que no son dominados por ningún otro y no pueden ser descartados ya que son potenciales caminos óptimos. Esto se debe a que, por como definimos la relación de dominancia, muchos pares de caminos no son comparables entre sí, es decir, que para muchos pares de caminos C_1 y C_2, no puede afirmarse que C_1 domine a C_2 ni que C_2 domine a C_1.

Con el objetivo de reducir el tiempo que toma encontrar el camino de costo mínimo en el grafo implementamos dos modificaciones en el algoritmo. La primera consiste en limitar la cantidad de rutas a analizar, generando sólo el subconjunto de las rutas que verifique determinadas restricciones. Es importante aclarar que, si bien al imponer estas restricciones se limita la cantidad de rutas a analizar y se reduce el tiempo de ejecución del algoritmo, también se reduce la cantidad de soluciones factibles.

La segunda modificación consiste en redefinir la relación de dominancia haciéndola menos estricta. De esta forma buscamos aumentar la cantidad de caminos dominados, ya que eso disminuye el espacio de caminos a analizar por el algoritmo y, en consecuencia, acelera su tiempo de ejecución. En los casos en los que el camino óptimo resultante no tiene costo negativo, volvemos a aplicar el algoritmo utilizando la definición original de dominancia.

Para aumentar la cantidad de soluciones factibles enteras que incluyan rutas generadas por el algoritmo de camino mínimo implementamos otras dos modificaciones. La primera consiste en agregar a la matriz A más de una columna por iteración, de forma que el método *Simplex* cuente con más columnas al momento de seleccionar una candidata a ingresar a la base. La segunda modificación consiste en generar, para cada ruta de costo mínimo, un conjunto de rutas que la *complemente*, de forma que la ruta de costo mínimo cuente con camioneros disponibles en las ciudades y en los momentos en los que los requiere. Esta última variante pretende contribuir a la satisfacibilidad de la ruta de costo óptimo mejorando la disponibilidad de camioneros en los lugares y momentos en que ésta los precise.

A continuación detallamos las modificaciones implementadas para reducir el tiempo de ejecución. La modificación concerniente con aumentar la cantidad de soluciones factibles, a la que llamaremos *generación de rutas complementarias*, se detallará en la próxima sección, ya que requiere varias definiciones y una descripción más extensa que el resto de las modificaciones implementadas.

5.1.4. Restricciones sobre rutas

La primer modificación que implementamos consiste en reducir la cantidad de rutas que el algoritmo debe considerar. Para ello, agregamos restricciones sobre los caminos, reduciendo de esta forma el espacio de rutas a generar. Estas restricciones están asociadas a las ciudades visitadas entre cualquier par de eventos consecutivos, y al tiempo que el camión permanece detenido en una ciudad.

Con respecto a las ciudades visitadas entre eventos, consideramos dos restricciones. La primera consiste en limitar la cantidad de ciudades por las que un camión puede pasar entre dos eventos. Por ejemplo, si la cantidad de ciudades a visitar se restringe a dos, un

camino que recorre tres ciudades entre el *pick-up* de un pedido y el *drop-off* del mismo no será considerado como un camino válido. Es decir, el algoritmo lo descartará, de la misma forma que descartaría al camino si el tiempo consumido por el mismo fuera mayor que el establecido por el horizonte de planificación, o si realizara el *pick-up* de un mismo pedido más de una vez. La segunda restricción asociada a las ciudades visitadas entre eventos consiste en impedir que un camión pase por una misma ciudad más de una vez entre dos eventos consecutivos. Dicho de otra manera, se fuerza a que la secuencia de ciudades visitadas entre eventos no contenga ciudades repetidas.

El otro tipo de restricción tiene en cuenta el tiempo que el camión asociado a la ruta pasa detenido en una ciudad. Dado que cada camión no está asociado a un único camionero, si un conductor llegó al límite de las horas permitidas de trabajo y necesita descansar, el camión puede seguir su recorrido con otra tripulación a bordo. Esto permite que los camiones estén en actividad muchas horas consecutivas, y que de esa manera sean utilizados para llevar a cabo la mayor cantidad posible de pedidos. Agregamos entonces la restricción de que, en cada nodo del camino, el porcentaje de tiempo que el camión estuvo detenido en alguna ciudad no supere cierto límite.

Es importante aclarar que las restricciones sobre la generación de rutas reducen el espacio de búsqueda del algoritmo, por lo que podría ocurrir que ciertas rutas que podían llegar a ser óptimas ya no sean consideradas válidas. Esto restringe la optimalidad del problema de acuerdo a su definición original.

5.1.5. Relajación de la relación de dominancia

Como mencionamos en la definición 5.3, para que un camino C_1 domine a otro camino C_2, ambos deben haber consumido el mismo tiempo. Esta restricción hace que la cantidad de caminos no comparables entre sí sea muy grande. Recordemos que sólo aquellos caminos dominados por otros pueden ser descartados por el algoritmo de camino mínimo, y por ese motivo sería conveniente contar con una definición de dominancia menos estricta.

Con este objetivo definimos una nueva relación de dominancia modificando la condición respecto al tiempo de la definición original. Según esta nueva definición, para que un camino C_1 domine a otro C_2, no es necesario que ambos caminos consuman la misma cantidad de tiempo, sino que basta con que la diferencia entre el tiempo consumido por C_2 y el tiempo consumido por C_1 sea menor que cierta cota.

Definición 5.4 (Relación de dominancia relajada)
Sea $c \in \mathbb{N}_0$ una cota, se define la relación de dominancia relajada $\preccurlyeq_c^*$ *para la cota c de la misma forma que la definición original (definición 5.3) reemplazando la condición con respecto al tiempo de los caminos por la siguiente:*

$$0 \leq T(C_2) - T(C_1) \leq c$$

Notar que si $c = 0$, entonces la relación de dominancia relajada es igual a la relación de dominancia original.

Recordemos que la definición de dominancia garantizaba que si un camino C_1 domina a otro camino C_2, cualquier extensión válida de C_2 resulta también una extensión válida de C_1. Además, dado que C_1 tiene menor costo, cualquier extensión de C_1 tiene menor costo que la misma extensión aplicada sobre C_2. En la nueva definición de dominancia relajada se conserva la primera de estas propiedades pero no la segunda. Esto se debe a que el costo real de un camino es función no sólo de los ejes que lo componen, sino también del momento en que dichos ejes son transitados (ver definición 5.1).

Dado que, según la nueva definición, el tiempo consumido por C_2 debe ser mayor o igual que el tiempo consumido por C_1, si considerarámos una extensión de ambos caminos que incluya a un eje e, dicho eje podría no ser transitado en el mismo momento por la extensión de C_1 que por la extensión de C_2. A pesar de que el costo de C_1 es menor o igual al costo de C_2, no podemos determinar si el costo de la extensión de C_1 será menor que el costo de la misma extensión sobre C_2. Sin embargo, sucede que $C_1 \preccurlyeq_c^* C_2$ por lo que la variante del algoritmo descartará al camino C_2 para optar por C_1.

Siguiendo el anterior razonamiento podría ocurrir que alguna extensión del camino descartado C_2 eventualmente finalizara en el nodo *sink* y tuviera un costo real negativo, mientras que la misma extensión sobre C_1 finalizara con costo real positivo. Al finalizar la ejecución del algoritmo, se obtendría como resultado que no existe un camino de costo negativo.

Si bien la definición de dominancia relajada permite que el algoritmo descarte muchos caminos y finalice más rápido, ya no podemos garantizar que si el camino válido obtenido tiene costo positivo no existan caminos válidos de costo negativo. Es decir, el algoritmo modificado deja de ser un algoritmo exacto y se transforma en una heurística. En los casos en los cuales el algoritmo heurístico nos devuelva un camino de costo negativo, podremos utilizar la ruta generada aunque no tengamos la certeza que es la de costo menor. En caso contrario, utilizaremos el algoritmo exacto para verificar si existe algún camino válido de costo negativo.

Es importante notar que el comportamiento de la heurística depende fuertemente de la cota que se utilice en la relación de dominancia relajada, ya que este parámetro regula, de cierta manera, qué caminos serán descartados por el algoritmo. En el capítulo 7 se presentan los resultados obtenidos por la heurística para distintos valores de esta cota.

5.1.6. Rutas potencialmente beneficiosas

Con el objetivo de incrementar la cantidad de soluciones factibles enteras, modificamos el algoritmo para que devuelva no sólo la ruta de menor costo, sino todas aquellas rutas de costo negativo. De esta manera, al resolver cada nueva relajación, agregamos más de una columna, lo que permite que el método *Simplex* tenga más opciones para elegir la columna a ingresar a la base.

Llamamos *rutas potencialmente beneficiosas* a todas estas rutas de costo negativo, ya que cualquiera de ellas podría ser elegida por *Simplex* para formar parte de la solución de la iteración correspondiente. Recordemos que originalmente obteníamos la ruta de costo mínimo sólo para garantizar que, de existir, se agregara a la matriz A de restricciones una columna asociada a una variable de costo reducido negativo.

5.2. Columnas complementarias

El método principal de generación de columnas presentado en la sección 5.1.2 busca las mejores rutas en términos de su costo reducido. Cuando se aplican las restricciones que fuerzan a que las variables tomen valores enteros, puede suceder que muchas de estas columnas, que antes podían ser incluidas en la solución de la relajación, no puedan ser parte de la solución entera. Esto podría tener múltiples causas, pero consideramos que una de las más relevantes es la descoordinación entre los descensos y ascensos de camioneros. Con esto queremos decir que una ruta puede requerir un camionero en cierta ciudad y momento, y dicho requerimiento podría estar satisfecho en la relajación pero no así en el problema restringido a que las variables sean enteras.

Para intentar solucionar este problema desarrollamos la idea de generar rutas que complementen a un conjunto de rutas obtenidas por el método principal de generación. El objetivo de una ruta complementaria es hacer descender camioneros de manera que los mismos estén disponibles en las ciudades y momentos en los que las rutas del conjunto original los requieran. De esta manera, creemos que es más probable que las rutas con costo reducido negativo puedan ser incluidas en la solución del problema entero.

5.2.1. El problema de complementar rutas

Antes de presentar el problema de complementar un conjunto de rutas debemos introducir algunas definiciones.

Definición 5.5 (Función de ascenso de camioneros)
Dado un conjunto de rutas $S \subseteq R$, una ciudad k y un momento m, la función de ascensos de camioneros *suma todos los ascensos de camioneros en dicha ciudad y momento para el conjunto de rutas S.*

$$camAsc^* : \mathcal{P}(R) \times L \times M \to \mathbb{N}_0$$
$$camAsc^*(S, k, m) = \sum_{r \in S} camAsc(r, k, m)$$

donde $\mathcal{P}(R)$ denota el conjunto de partes de R.

Los ascensos de los camioneros del conjunto de rutas definen los requerimientos de camioneros que tienen estas rutas en una determinada ciudad y momento. Estos requerimientos de camioneros pueden estar satisfechos o insatisfechos dependiendo de si en esa ciudad hay camioneros disponibles o no. Más adelante definiremos la noción de disponibilidad de camioneros que nos ayudará a formalizar esta idea.

Definición 5.6 (Función de descenso de camioneros)
Dado un conjunto de rutas $S \subseteq R$, una ciudad k y un momento m, la función de descensos de camioneros *suma todos los descensos de camioneros en dicha ciudad y momento para el conjunto de rutas S.*

$$camDesc^* : \mathcal{P}(R) \times L \times M \to \mathbb{N}_0$$
$$camDesc^*(S, k, m) = \sum_{r \in S} camDesc(r, k, m)$$

Definición 5.7 (Función de disponibilidad de camioneros)
Dado un conjunto de rutas $S \subseteq R$, una ciudad k y un momento m se define la función de disponibilidad de camioneros *como:*

$$dispCam : \mathcal{P}(R) \times L \times M \to \mathbb{N}_0$$
$$dispCam(S, k, m) = \begin{cases} |DCL_k| - camAsc^*(S, k, 0) & si\ m = 0 \\ dispCam(S, k, m-1) + camDesc^*(S, k, m) - reqSat(S, c, m) & si\ m > 0 \end{cases}$$

donde DCL_k es el conjunto de camioneros que comienzan su itinerario en la ciudad k y $reqSat$ es la función de requerimientos satisfechos de camioneros que se definirá a continuación.

En un comienzo, la disponibilidad de camioneros en una ciudad está dada por la cantidad de camioneros que se encuentran allí inicialmente descontando aquellos que ascienden a algún camión. Luego, la disponibilidad de camioneros para una determinada ciudad y momento depende de la disponibilidad en dicha ciudad en el momento anterior y de los descensos y ascensos de camioneros en ese momento.

Si un camionero desciende en una ciudad queda disponible en la misma hasta que ascienda a un camión en algún momento posterior. De esta manera, la disponibilidad aumenta cuando se produce el descenso de un camionero. Por otro lado, la disponibilidad disminuye cuando una ruta requiere de un camionero (i.e. se produce el ascenso de un camionero).

A partir de conocer la disponibilidad de camioneros en una ciudad y momento determinados, podemos saber cuántos requerimientos (o ascensos) de camioneros pueden ser satisfechos y cuántos no. Definimos entonces las nociones de requerimientos de camioneros satisfechos e insatisfechos para terminar de formalizar el problema.

Definición 5.8 (Función de requerimientos satisfechos de camioneros)
Dado un conjunto de rutas $S \subseteq R$, una ciudad k y un momento m, se define la función de requerimientos satisfechos de camioneros *como:*

$$reqSat : \mathcal{P}(R) \times L \times M \to \mathbb{N}_0$$

$$reqSat(S, k, m) = min(camAsc^*(S, k, m), dispCam(S, k, m-1) + camDesc^*(S, k, m))$$

Definición 5.9 (Función de requerimientos insatisfechos de camioneros)
Dado un conjunto de rutas $S \subseteq R$, una ciudad k y un momento m, definimos la función de requerimientos insatisfechos *en base a los ascensos de camioneros y cuántos de ellos fueron satisfechos.*

$$reqInsat : \mathcal{P}(R) \times L \times M \to \mathbb{N}_0$$

$$reqInsat(S, k, m) = camAsc^*(S, k, m) - reqSat(S, k, m)$$

En base a estas definiciones podemos construir una función δ que nos permite medir cuánto mejora la insatisfacibilidad de requerimientos de camioneros de un conjunto de rutas al agregar una nueva ruta a dicho conjunto. A continuación definimos dicha función.

Definición 5.10 (Función de mejora de insat. de requerimientos)
Dado un conjunto de rutas $S \subseteq R$ y una ruta $r \in R$ tal que $r \notin S$, definimos la función de mejora de insatisfacibilidad de requerimientos de camioneros *como:*

$$\delta : \mathcal{P}(R) \times R \to \mathbb{Z}$$

$$\delta(S, r) = \sum_{c \in L} \sum_{m \in M} reqInsat(S, c, m) - \sum_{c \in L} \sum_{m \in M} reqInsat(S \cup \{r\}, c, m)$$

El problema de complementar un conjunto de rutas S consiste en encontrar aquella ruta r que maximice la función $\delta(S, r)$, de manera que el conjunto de rutas resultante $(S \cup \{r\})$ tenga la menor cantidad posible de requerimientos insatisfechos. Esta manera de definir el problema nos permite ir complementando conjuntos en forma iterativa, es decir, obtener la mejor ruta complementaria para un conjunto de rutas, agregarla a dicho conjunto, luego complementar el nuevo conjunto, y así sucesivamente.

Puede suceder que, para un conjunto dado de rutas S, la ruta r que maximice $\delta(S, r)$ produzca que el valor de dicha función no sea positivo (i.e. $\delta(S, r) \leq 0$). Esto quiere

decir que la mejor ruta que se pudo generar no contribuye a mejorar los requerimientos insatisfechos de camioneros de S, sino que ocurre lo contrario. En este caso no tiene sentido que se la considere para ser agregada al problema.

5.2.2. Resolución del problema de complementar rutas

Dado un conjunto de rutas S, para encontrar la ruta r que maximice la función δ utilizamos una variante del algoritmo para resolver el RCSPP descripto en la sección 5.1.2. En la versión original del algoritmo, cada camino contaba con su costo acumulado y dicho costo se utilizaba como criterio en la relación de dominancia para establecer si un camino era mejor que otro. La variante consiste en utilizar como criterio de decisión la mejora acumulada de requerimientos insatisfechos de un camino. Para esto definimos una nueva relación de dominancia que considera la mejora en los requerimientos insatisfechos de camioneros para el conjunto S.

Definición 5.11 (Rel. de dominancia por mejora de insat. de requerimientos) *Dado un conjunto de rutas $S \subseteq R$ y dos caminos C_1, C_2 del grafo GR asociados a rutas r_1, r_2 respectivamente, C_1 domina por mejora de insatisfacibilidad de requerimientos a C_2 ($C_1 \trianglelefteq_S C_2$) sii*

- *C_1 y C_2 finalizan en el mismo nodo*
- *La mejora en los requerimientos insatisfechos de camioneros (con respecto a S) producida por C_1 es mayor o igual a la producida por C_2 : $\delta(S, r_1) \geq \delta(S, r_2)$*
- *El tiempo de C_1 es menor o igual al de C_2 : $T(C_1) \leq T(C_2)$*
- *Si ambos caminos tienen pendiente el drop-off del último pedido:*
  - *$ultimo(P(C_1)) = ultimo(P(C_2))$*

  Si no
  - *Tanto C_1 como C_2 no deben tener pendiente el drop-off del último pedido realizado*
- *Los pedidos realizados por C_1 están incluidos en los de C_2 : $P(C_1) \subseteq P(C_2)$*

Utilizando esta nueva definición de dominancia, el algoritmo nos devolverá un camino cuya ruta asociada, de ser incluida en el conjunto actual S, mejora la cantidad de requerimientos insatisfechos. La idea es que al agregar la ruta asociada al camino resultante al conjunto S, el conjunto de rutas quede más equilibrado en términos de la coordinación entre los ascensos y descensos de camioneros.

En el capítulo 6 veremos cómo se introduce la generación de rutas complementarias en el algoritmo de resolución general y en el capítulo 7 analizaremos si la generación de estas rutas contribuye a mejorar la solución obtenida.

5.3. Columnas iniciales

Como mencionamos en el capítulo 2, el método *Simplex* explora distintas soluciones factibles del problema, mejorando en cada iteración el valor de la función objetivo. Al utilizar la técnica de generación de columnas, no contamos en todas las iteraciones de *Simplex* con la totalidad de las variables del problema. En particular, en la primera iteración no se

cuenta con ninguna variable correspondiente a rutas (y_r) de manera que *Simplex* encuentre una solución factible para comenzar. Es por esto que es necesario generar un conjunto de rutas iniciales que verifiquen todas las restricciones asociadas a variables de ruta antes de comenzar el proceso de generación de columnas.

A continuación presentamos el pseudocódigo del algoritmo implementado para obtener un conjunto de rutas iniciales.

Algoritmo 5 Generación de conjunto de rutas iniciales

Input: CCL_k camiones que comienzan en ciudad k $\forall k \in L$, DCL_k camioneros que comienzan en ciudad k $\forall k \in L$, P conjunto de pedidos, L conjunto de ciudades
Output: RI el conjunto de rutas iniciales

1: $RI \leftarrow \emptyset$
2: **for all** ciudad $k \in L$ **do**
3: $cant \leftarrow min(|CCL_k|, |DCL_k|)$
4: Agregar a RI $cant$ comienzos de ruta a partir de la ciudad k
5: **end for**
6: *// Para cada pedido, revisar a qué ruta conviene asignárselo.*
7: **for all** pedido $p \in P$ **do**
8: Buscar ruta $r \in RI$ que mejor realice el pedido p
9: Extender ruta r para que realice el pedido p
10: **end for**
11: **return** RI

Las rutas generadas por este algoritmo tendrán la particularidad de que cada camión será manejado por un único camionero a lo largo de todo el recorrido. Para que un camionero pueda comenzar una ruta en un camión, ambos deben encontrarse en la misma ciudad en el momento inicial. Es por esto que la cantidad de rutas que pueden comenzar en cada ciudad queda determinada por la cantidad de camiones y camioneros que se encuentran en esa ciudad en el momento inicial, como puede verse en los pasos 3 y 4.

Una vez que determinamos con cuántas rutas vamos a contar, debemos definir qué pedidos realizará cada una de ellas. Para esto, le asignamos a cada pedido aquella ruta que *mejor* lo realiza. En este contexto, la ruta que mejor realiza un pedido es aquella que puede realizar su correspondiente *drop-off* lo antes posible.

Para determinar qué ruta del conjunto realizará el pedido en forma completa en el menor tiempo calculamos cuánto demora cada una de ellas en realizar el *pick-up* del pedido y luego su correspondiente *drop-off*. Una vez calculado el tiempo de finalización de *drop-off* para cada ruta, nos quedamos con aquella que lo finalizó antes. Para calcular el tiempo de finalización de un evento, ya sea *pick-up* o *drop-off*, utilizamos el algoritmo 6.

Dados una ruta, un pedido y un evento (*pick-up* o *drop-off*), el algoritmo calcula el momento más temprano en el que dicha ruta puede realizar el evento asociado a ese pedido. Para esto, debe considerar cuánto demora el camión en llegar a la ciudad donde debe realizar el evento, la cantidad de horas de trabajo consecutivo que acumula el camionero que lo llevará a cabo y si llega a la ciudad de destino dentro de la correspondiente ventana de tiempo. Todas estas condiciones son necesarias para que se cumplan las restricciones del problema.

A continuación describimos brevemente las funciones utilizadas en el pseudocódigo del algoritmo 6.

- $momentoUltimoEvento(r)$: indica en qué momento se realizó el último evento (*pick-up* o *drop-off*) asociado a la ruta en cuestión.

Algoritmo 6 Cálculo del momento en que una ruta finaliza un evento

Input: Ruta r, pedido p, evento e (*pick-up* o *drop-off*)
Output: El momento m en el cual la ruta r finaliza el evento e del pedido p

```
1: m ← momentoUltimoEvento(r)
2: // Si no está en la ciudad donde debe realizar el evento, debe viajar
3: if ciudadActual(r) ≠ ciudadEvento(p, e) then
4:     // Verificar si debe tomarse franco o descansar
5:     if necesitaDescansoOFranco(r, p) then
6:         Calcular el mínimo tiempo td que necesita descansar
7:         m ← m + td
8:     end if
9:     // Viajar hasta la ciudad del evento
10:    m ← m + tiempo(ciudadActual(r), ciudadEvento(p, e))
11: end if
12: if m no está dentro de la ventana de tiempo diaria de p then
13:    Calcular tiempo de espera te hasta que pueda realizar el evento e
14:    m ← m + te
15: end if
16: // Realizar el evento, lo que demora K/24 momentos
17: m ← m + K/24
18: return m
```

- $ciudadActual(r)$: indica cuál fue la última ciudad visitada por la ruta. Es decir, en qué ciudad se encuentra actualmente el camión asociado a la ruta analizada.
- $ciudadEvento(p, e)$: indica la ciudad donde debe realizarse el evento correspondiente al pedido.
- $necesitaDescansoOFranco(r, p)$: indica si el camionero asociado a la ruta en cuestión puede llevar a cabo el viaje necesario para realizar cierto evento, teniendo en cuenta las restricciones respecto de la cantidad de horas consecutivas de trabajo y de la cantidad de días de franco.
- $tiempo(c_1, c_2)$: devuelve el tiempo que demanda el viaje entre dos ciudades.

El algoritmo 5 tiene como desventaja que no contempla el conjunto de pedidos en su totalidad, sino que, de manera golosa, asigna a cada pedido la ruta que antes lo pueda llevar a cabo. Esto podría traer como consecuencia que, después de definir qué rutas realizarán los primeros pedidos, algún pedido no pueda ser realizado por ninguna de las rutas disponibles dentro del horizonte de tiempo establecido.

Como todos los pedidos deben ser realizados por algún camión, el conjunto de rutas generado en ese caso no formaría parte de una solución factible. Estos casos pueden salvarse ejecutando el algoritmo nuevamente, modificando el orden en el que se iteran los pedidos, ya que al recorrerlos en otro orden se podría encontrar un conjunto de rutas que los satisfaga a todos. En nuestra implementación, este algoritmo se ejecuta dos veces: la primera recorriendo los pedidos en orden ascendente y la segunda en el orden inverso. A pesar de esto, la limitación de considerar sólo aquellas rutas en las que cada camión sea manejado por un único camionero podría llevar a que no se puedan satisfacer todas las restricciones, incluso aunque existiera una solución factible al problema. En la práctica, sin embargo, el algoritmo propuesto encontró un conjunto de rutas que verifican todas las condiciones en la mayoría de los casos.

Capítulo 6

Algoritmo de resolución general

En los capítulos anteriores describimos tanto el grafo de rutas como el algoritmo que utilizamos para encontrar un camino de costo mínimo dentro del mismo. Describimos también los algoritmos para generar un conjunto inicial de rutas que satisfagan todos los pedidos a realizar, y las variaciones que desarrollamos para limitar la cantidad de soluciones a explorar. En este capítulo describiremos cómo se relacionan todos estos algoritmos para resolver el VCSPDPTW, y mostraremos el pseudocódigo del algoritmo de resolución general implementado.

6.1. Algoritmo para el VCSPDPTW

Como mencionamos en el capítulo 5, el modelo propuesto cuenta con una gran cantidad de variables de ruta (y_r). Dado que en la práctica es imposible considerar todas estas variables en forma explícita, decidimos utilizar la técnica de generación de columnas para resolver el problema. Este esquema de resolución plantea dividir al problema en un problema maestro y un subproblema de generación de columnas, en donde las variables de ruta serán generadas en cada iteración del problema maestro.

En el algoritmo 7 se muestra el esquema de resolución general para el VCSPDPTW. El algoritmo comienza armando un conjunto de columnas iniciales para que el método *Simplex* cuente con una solución factible inicial con la que pueda comenzar. Este conjunto de columnas iniciales es el generado por el algoritmo 5 presentado en el capítulo anterior. El siguiente paso consiste en construir la formulación del modelo cuya matriz de restricciones contiene únicamente las columnas asociadas a las variables explícitas en el modelo ($X_{dki}, Y_{dki}, W_{dki}, Z_{dj}$) y aquellas columnas iniciales recién generadas. Es importante aclarar que esta formulación representa la relajación lineal de la presentada en la sección 3.2.

Una vez hecho esto se procede a generar y agregar nuevas columnas a la formulación relajada. Este procedimiento, que será detallado en la siguiente sección, es un procedimiento iterativo que consiste en aplicar el método *Simplex* para encontrar el óptimo de la relajación del problema con las columnas con las que se cuenta hasta el momento, y luego generar una o más columnas para agregar a la formulación y reoptimizar. Este proceso se repite hasta que se alcanza cierta cantidad de iteraciones o se alcanza cierta cota de tiempo.

Cuando concluye la etapa de generación de columnas sobre la relajación, se construye una nueva formulación en base a la resultante del proceso de generación de columnas, pero con la restricción de que las variables sean binarias. Esta es la formulación original del modelo que presentamos en la sección 3.2, con la diferencia de que en vez de considerar las

Algoritmo 7 Resolución general de VCSPDPTW

Input: Datos del problema *dp*, *cmi* la cantidad máxima de iteraciones, *tme* el tiempo máximo de ejecución, *rpb* si se deben considerar rutas potencialmente beneficiosas, *crc* la cantidad de rutas a complementar, *crcg* la cantidad de rutas complementarias a generar

Output: x^* solución óptima

1: $RI \leftarrow generarRutasIniciales(dp)$
2: *// Construir la formulación sólo con las columnas iniciales*
3: Sea PPL la r.l. de formulación del VCSPDPTW considerando sólo las rutas $r \in RI$
4: *// Generar columnas para PPL*
5: $generarYAgregarColumnas(PPL, dp, cmi, tme, rpb, crc, crcg)$
6: Sea $PPLB$ la formulación asociada a PPL restringida a variables binarias
7: Resolver $PPLB$ utilizando *Branch and Cut*

variables correspondientes a todas las posibles rutas, sólo estarán presentes las que fueron generadas en la etapa anterior. Luego se aplica *Branch and Cut* (ver sección 2.3.3) sobre la formulación resultante para obtener una solución entera.

Utilizamos la implementación de *Branch and Cut* provista por el paquete CPLEX®, la cual genera cortes de uso general como los de Gomory, Clique, GUB (*Generalized Upper Bound*), etc [IBM10]. El paquete CPLEX® permite configurar muchos de los parámetros del algoritmo de *Branch and Cut* aunque nosotros decidimos utilizar los valores por defecto.

El algoritmo propuesto para la resolución del VCSPDPTW se asemeja a un esquema de *Branch and Price* (ver sección 2.3.4). La diferencia radica en que un método de *Branch and Price* utiliza la técnica de generación de columnas para resolver las relajaciones lineales asociadas a cada uno de los nodos resultantes del proceso de *branching*, mientras que nuestro algoritmo sólo la utiliza al momento de resolver la relajación inicial, es decir, la del nodo raíz del árbol. Esto fue hecho en forma deliberada, ya que el alcance del trabajo no incluía analizar el impacto que tenían las estrategias de *branching* en el, ya de por sí complicado, método de generación de columnas.

De esta manera, el algoritmo presentado resulta ser una heurística y no un algoritmo exacto. Esto es así dado que el criterio de parada de un algoritmo exacto de estas características depende de poder aseverar que no existan más columnas que puedan mejorar el valor de la función objetivo. Para poder asegurar esto es necesario realizar el proceso de generación de columnas en cada uno de los nodos del árbol y no sólamente en el nodo raíz.

6.2. Algoritmo para generación de columnas

El algoritmo 8 muestra el procedimiento de generación de columnas y agregado de las mismas a la formulación relajada. Comienza construyendo el grafo de rutas a partir de los datos de entrada del problema y luego ingresa en el ciclo principal de generación de columnas. Aquí se resuelve la relajación del problema utilizando la implementación de *Simplex* provista en el paquete CPLEX®. Una vez resuelta dicha relajación se obtienen los valores de la solución al problema dual asociado y, con los mismos, se actualizan los costos de los ejes en el grafo de rutas como describimos en la sección 5.1.1.

Para generar rutas de costo reducido negativo utilizamos el algoritmo para el RCSPP descripto en 5.1.2 así como las variantes presentadas. Este algoritmo es exacto y analiza una cantidad muy grande de caminos, por lo que si el problema cuenta con muchas ciudades y pedidos, es posible que demore mucho tiempo en encontrar el camino de menor costo.

Es por esto que inicialmente invocamos la variante del algoritmo que utiliza la relación de dominancia relajada (ver definición 5.4). Esto puede verse en la línea 9 del pseudocódigo.

Como ya mencionamos, esta modificación permite descartar muchos caminos al momento de buscar el camino óptimo, de forma que se reduce considerablemente el tiempo de ejecución del algoritmo. La desventaja de esta modificación es que el algoritmo deja de ser exacto y se transforma en una heurística. Esto quiere decir que el hecho de que el algoritmo no encuentre un camino de costo negativo no implica que dicho camino no exista. En los casos en los que ocurre esto, invocamos al algoritmo utilizando la definición original de dominancia (ver definición 5.3). En estos casos el algoritmo demora más tiempo en devolver la solución, pero si el camino resultante tiene costo positivo se tiene la certeza de que no existe ningún camino de costo negativo y puede finalizarse el proceso de generación de columnas.

Si se encuentra un camino de costo negativo, ya sea mediante la heurística o mediante el algoritmo exacto, debemos construir la columna asociada a dicho camino e incluirla en la matriz de restricciones de la formulación. En un esquema de generación de columnas típico, una vez hecho esto se procedería a la siguiente iteración. Sin embargo, como mencionamos en la sección 5.1.3, implementamos dos variantes que permiten agregar más de una columna por iteración.

La primera de estas variantes consiste en agregar todas las columnas de costo reducido negativo en lugar de agregar sólo aquella de menor costo. Recordemos que llamamos a estas columnas *potencialmente beneficiosas*, ya que cualquiera de las mismas podría formar parte de la solución en la siguiente iteración. En las líneas 23 a 26 del pseudocódigo puede verse que, en caso de que el parámetro correspondiente así lo indique, se agregan las columnas potencialmente beneficiosas de la misma forma en que se agrega la columna de costo mínimo. El hecho de tener o no en cuenta estas columnas está parametrizado ya que, a priori, no queda claro que se obtengan mejores resultados si se las utiliza. Por este motivo, realizamos pruebas en las que se agregan las columnas potencialmente beneficiosas, y otras pruebas en las que en cada iteración agregamos únicamente la columna correspondiente al camino de costo mínimo. En el próximo capítulo presentaremos las pruebas realizadas y los resultados obtenidos.

El segundo mecanismo que implementamos para no agregar una única columna por iteración es la generación de rutas complementarias descripta en la sección 5.2. A medida que se van agregando rutas al problema, las mismas se van almacenando en un conjunto que llamaremos *conjunto de rutas a complementar*. En dicho conjunto se incluyen tanto las rutas de costo mínimo como las rutas potencialmente beneficiosas que van a ser agregadas a la formulación del problema (estas últimas sólo en el caso de estar habilitada la generación de las mismas). Cuando el tamaño del conjunto alcanza cierto valor determinado por medio de un parámetro, se generan las rutas que complementan a las del conjunto. La cantidad de rutas complementarias que se generan también está dada por un parámetro. Una vez generadas las rutas complementarias, las mismas se agregan al problema y se vacía el conjunto de rutas a complementar. Las columnas de costo mínimo y potencialmente beneficiosas que se agregan en iteraciones posteriores van conformando el nuevo conjunto de rutas a complementar, y una vez que dicho conjunto alcanza el tamaño indicado se repite este proceso.

Dado que no sabíamos si las rutas complementarias mejorarían los resultados obtenidos, la utilización de las mismas está sujeta a un parámetro. De esta forma, debe ocurrir que tanto la cantidad de rutas a complementar como la cantidad de rutas complementarias a generar sean positivas.

Algoritmo 8 Generación de columnas

Input: La formulación del modelo PPL, los datos del problema dp, la cantidad máxima de iteraciones cmi, tme el tiempo máximo de ejecución, rpb si se deben considerar rutas potencialmente beneficiosas, crc la cantidad de rutas a complementar, $crcg$ la cantidad de rutas complementarias a generar

Output: Agrega a PPL las columnas correspondientes según los parámetros de entrada

1: $GR \leftarrow construirGrafoRutas(dp)$
2: *// El conjunto de rutas a complementar*
3: $RAC \leftarrow \emptyset$
4: $tiempo \leftarrow 0, cantIteraciones \leftarrow 0$
5: *// Mientras se deba seguir iterando y el tiempo transcurrido no alcance el límite*
6: **while** $cantIteraciones < cmi \wedge tiempo < tme$ **do**
7: Resolver PPL utilizando *Simplex*
8: Sea y^* el vector solución del problema dual asociado a PPL
9: $actualizarCostosEjes(GR, y^*)$
10: Obtener el camino mínimo C^* en GR y el conjunto RPB de todos los caminos válidos de costo negativo utilizando el algoritmo heurístico
11: *// Si el costo del camino mínimo es positivo*
12: **if** $\Sigma(C^*) \geq 0 \wedge tiempo < tme$ **then**
13: Obtener el camino mínimo C^* en GR y el conjunto RPB de todos los caminos válidos de costo negativo utilizando el algoritmo exacto
14: *// Si no hay caminos de costo negativo, terminar*
15: **if** $\Sigma(C^*) \geq 0$ **then**
16: **break**
17: **end if**
18: **end if**
19: Construir columna $A_{y_{c^*}}$ correspondiente C^*
20: Agregar $A_{y_{c^*}}$ a PPL
21: *// Agregar el camino al conjunto de rutas a complementar*
22: $RAC \leftarrow RAC \cup \{C^*\}$
23: *// Si corresponde, agregar todas las columnas de costo reducido negativo*
24: **if** rpb **then**
25: Construir columnas A_{y_r} correspondientes a los caminos en RPB
26: Agregar columnas A_{y_r} a PPL
27: **end if**
28: *// Si corresponde, generar rutas complementarias y agregarlas a la formulación*
29: **if** $crc > 0 \wedge |RAC| = crc \wedge crcg > 0$ **then**
30: Generar CRC conjunto de $crcg$ rutas complementarias para RAC
31: Construir columnas A_{y_r} correspondientes a los caminos en CRC
32: Agregar columnas A_{y_r} a PPL
33: **end if**
34: $cantIteraciones \leftarrow cantIteraciones + 1$
35: $actualizarTiempo(tiempo)$
36: **end while**

El criterio de parada del ciclo principal está dado por la cantidad de iteraciones y el tiempo máximo de ejecución, ambos especificados por sus correspondientes parámetros. Si la cantidad de iteraciones realizadas o el tiempo transcurrido superan los límites establecidos, el algoritmo termina y concluye la etapa de generación de columnas.

Capítulo 7

Resultados

En este capítulo presentamos los resultados obtenidos utilizando el algoritmo introducido en el capítulo anterior. Dado que el algoritmo de resolución general es parametrizable con un amplio número de opciones, decidimos realizar algunas pruebas preliminares de forma de poder acotar el espectro de opciones para luego realizar las pruebas finales.

Los algoritmos fueron implementados en C++, utilizando la API de C del paquete optimizador CPLEX® versión 12.2.

7.1. Instancias del problema

Si bien este trabajo trata sobre un problema real de una compañía productora de café, no contábamos con datos reales con los que poder construir instancias de prueba, por lo que tuvimos que crear nuestras propias instancias del problema en forma manual. Para esto utilizamos información geográfica real de ciudades obteniendo las distancias que las separan por medio de *Google Maps*[1]. Dado que las distancias entre ciudades deben estar expresadas en unidad de tiempo, utilizamos las estimaciones de tiempo de viaje provistas por *Google Maps*, basadas en los límites de velocidad de cada segmento de ruta. El resto de la información de cada instancia del problema, como la distribución inicial de camiones y camioneros en el mapa, ventanas de tiempo, penalidad asociada a la entrega fuera de fecha de los pedidos y costo de viaje por unidad de tiempo, fueron configuradas en forma arbitraria.

Antes de determinar esta información es preciso decidir la cantidad de ciudades y de pedidos de la instancia del problema, y con cuántos camiones y camioneros se cuenta. En la solución propuesta, el aspecto que consideramos crítico con respecto a la *performance* es el tamaño del grafo de rutas, ya que el mismo impacta directamente sobre el desempeño del algoritmo utilizado para la generación de columnas. Como mostramos en el capítulo 4, el tamaño del grafo depende mayormente de la cantidad de ciudades y de pedidos. Es por esto que decidimos medir el tamaño de una instancia del problema en función de dichas cantidades. Las cantidades de camiones y camioneros en cada instancia fueron fijadas en función a la cantidad de ciudades: aproximadamente un camión por ciudad y dos camioneros por cada camión.

Con respecto al horizonte de planificación, medido en días, utilizamos el menor valor posible para cada instancia. Este parámetro también influye sobre el tamaño del grafo de rutas, por lo que acotarlo lo más posible también resulta beneficioso para la etapa de generación de columnas. El límite para el ajuste de este parámetro está dado por el

[1] http://maps.google.com

algoritmo de generación de rutas iniciales, dado que el hecho de que el mismo encuentre un conjunto de rutas que satisfagan todos los pedidos depende del horizonte establecido. Si el horizonte de planificación no es lo suficientemente grande, el algoritmo podría no encontrar una solución inicial, lo cual no permitirá obtener una solución al problema.

7.2. Pruebas preliminares

Las pruebas preliminares se realizaron sobre seis instancias del problema, con una cantidad de ciudades variando entre 6 y 7, y una cantidad de pedidos variando entre 6 y 12. Decidimos utilizar instancias de este tamaño, que pueden ser resueltas en un tiempo razonable, para poder fijar algunos de los parámetros del algoritmo y luego realizar pruebas sobre instancias más grandes. Realizamos estas pruebas preliminares en una PC con procesador Intel® Core™ i3 2.13 GHz y 4 GB de RAM.

7.2.1. Parámetros

Para organizar las pruebas, agrupamos los parámetros del algoritmo de resolución general en 4 categorías correspondientes a las variaciones implementadas sobre el algoritmo de generación de columnas descriptas en la sección 5.1.3. Las categorías son las siguientes:

- Restricciones sobre la generación de rutas
- Cota superior para dominancia relajada
- Rutas potencialmente beneficiosas
- Rutas complementarias

En la categoría de restricciones sobre la generación de rutas agrupamos a los parámetros *máxima cantidad de ciudades visitadas entre eventos* (MCCV), *ciudades visitadas entre eventos repetidas* (CVR) y *máximo porcentaje camión detenido en ciudad* (MPCD). Para esta categoría consideramos las siguientes tres alternativas:

- Sin restricción: MCCV: Sin límite (∞), CVR: sí, MPCD: 100 %
- Medianamente restringida: MCCV: 3, CVR: sí, MPCD: 50 %
- Muy restringida: MCCV: 1, CVR: no, MPCD: 20 %

La categoría de cota superior para dominancia relajada cuenta con un único parámetro, que es la cota que indica cuánta diferencia puede haber entre los tiempos acumulados por dos caminos para que uno de ellos pueda dominar al otro. Para este parámetro tuvimos en cuenta las siguientes opciones:

- Muy poco relajada: Cota = 1
- Poco relajada: Cota = 3
- Relajada: Cota = 5
- Totalmente relajada: Cota = ∞

La categoría de rutas potencialmente beneficiosas también está compuesta por un único parámetro. Dicho parámetro (RPB) indica si deben tenerse en cuenta o no las rutas potencialmente beneficiosas descriptas en la sección 5.1.6. Es decir, los posibles valores para este parámetro son verdadero (si se deben considerar las rutas potencialmente beneficionsas) y falso (si no deben ser consideradas).

En la última categoría de parámetros agrupamos a los parámetros relativos a las rutas complementarias, es decir, *cantidad de rutas a complementar* (CRC) y *cantidad de rutas complementarias a generar* (CRCG). Para esta categoría consideramos las siguientes alternativas:

- No se consideran rutas complementarias: CRC: 0, CRCG: 0
- Se consideran rutas complementarias: CRC: 5, CRCG: 10

A cada opción de los parámetros de una categoría la combinamos con todas las opciones de las otras categorías, lo que resultó en 48 posibles combinaciones para cada una de las instancias de prueba. En la tabla 7.1 se pueden ver todas las combinaciones de parámetros utilizadas.

7.2.2. Experimentos computacionales

A continuación presentamos un resumen de los resultados correspondientes a las corridas de las pruebas preliminares. Las descripciones detalladas de cada instancia, así como las tablas con todos los resultados que obtuvimos, se encuentran en el apéndice A. En estas pruebas preliminares limitamos el tiempo para la etapa de generación de columnas a 10 minutos, el tiempo para la etapa de resolución a 5 minutos y la cantidad de iteraciones del algoritmo de resolución general a 50 para todas las instancias.

Instancia 1

La instancia 1 consiste de 6 ciudades, 10 pedidos, 7 camiones y 13 camioneros. Los detalles de los pedidos, así como las distancias entre las ciudades y la ubicación inicial de los camiones y los camioneros pueden verse en la sección A.1.1. En la tabla A.4 se muestran los resultados para las corridas, correspondientes a las 48 diferentes configuraciones de parámetros. En esta instancia se obtuvieron mejoras de la solución inicial en 22 de las 48 configuraciones, las cuales se encuentran resaltadas en la tabla.

En general se mejoró la solución en aquellos casos donde se restringió la generación de rutas, ya sea en forma moderada (configuraciones 12, 13 y 33 a 38) o fuerte (configuraciones 17 a 19, 21, 22, 24 y 42 a 48). También observamos que la mejor solución encontrada, con valor 107, se encuentra siempre dentro de alguna configuración con generación de rutas muy restringidas. Solamente uno de los 22 casos de mejora se presenta en una configuración en la cual no se restringió la generación de rutas.

En cuanto a la cota superior para dominancia relajada, sólo en 5 de los 22 casos de mejora la cota es infinito. En los 17 casos restantes la cota estaba ajustada a 5, 3 o 1 (configuraciones 5, 12, 13, 19, 21, 22, 24, 35 a 38 y 43 a 48), por lo que parecería que poner un valor para la cota da buenos resultados. Por otro lado, observamos que con un valor de 5 para la cota ya se obtienen mejoras e inclusive hay casos en donde la solución mejora con un valor de 5 o de 3, pero no con un valor de 1 (por ejemplo como ocurre con las configuraciones 19, 21 y 23 respectivamente).

En 12 casos de mejora no se tuvieron en cuenta las rutas potencialmente beneficiosas mientras que en los 10 restantes sí fueron utilizadas. Si comparamos los pares de config-

Configuración	Restricciones rutas					Rutas complementarias	
	MCCV	CVR	MPCD	Cota	RPB	CRC	CRCG
1	∞	SI	100 %	∞	NO	0	0
2	∞	SI	100 %	∞	NO	5	10
3	∞	SI	100 %	5	SI	0	0
4	∞	SI	100 %	5	SI	5	10
5	∞	SI	100 %	3	SI	0	0
6	∞	SI	100 %	3	SI	5	10
7	∞	SI	100 %	1	SI	0	0
8	∞	SI	100 %	1	SI	5	10
9	3	SI	50 %	∞	NO	0	0
10	3	SI	50 %	∞	NO	5	10
11	3	SI	50 %	5	SI	0	0
12	3	SI	50 %	5	SI	5	10
13	3	SI	50 %	3	SI	0	0
14	3	SI	50 %	3	SI	5	10
15	3	SI	50 %	1	SI	0	0
16	3	SI	50 %	1	SI	5	10
17	1	NO	20 %	∞	NO	0	0
18	1	NO	20 %	∞	NO	5	10
19	1	NO	20 %	5	SI	0	0
20	1	NO	20 %	5	SI	5	10
21	1	NO	20 %	3	SI	0	0
22	1	NO	20 %	3	SI	5	10
23	1	NO	20 %	1	SI	0	0
24	1	NO	20 %	1	SI	5	10
25	∞	SI	100 %	∞	SI	0	0
26	∞	SI	100 %	∞	SI	5	10
27	∞	SI	100 %	5	NO	0	0
28	∞	SI	100 %	5	NO	5	10
29	∞	SI	100 %	3	NO	0	0
30	∞	SI	100 %	3	NO	5	10
31	∞	SI	100 %	1	NO	0	0
32	∞	SI	100 %	1	NO	5	10
33	3	SI	50 %	∞	SI	0	0
34	3	SI	50 %	∞	SI	5	10
35	3	SI	50 %	5	NO	0	0
36	3	SI	50 %	5	NO	5	10
37	3	SI	50 %	3	NO	0	0
38	3	SI	50 %	3	NO	5	10
39	3	SI	50 %	1	NO	0	0
40	3	SI	50 %	1	NO	5	10
41	1	NO	20 %	∞	SI	0	0
42	1	NO	20 %	∞	SI	5	10
43	1	NO	20 %	5	NO	0	0
44	1	NO	20 %	5	NO	5	10
45	1	NO	20 %	3	NO	0	0
46	1	NO	20 %	3	NO	5	10
47	1	NO	20 %	1	NO	0	0
48	1	NO	20 %	1	NO	5	10

Tabla 7.1: Combinaciones de parámetros para pruebas preliminares

uraciones que difieren en el uso de estas rutas, tenemos que en 5 casos (configuraciones 17 vs 41, 11 vs 35, 14 vs 38, 20 vs 44 y 23 vs 47) la mejora se da cuando no se utilizan, en 3 casos (configuraciones 5 vs 29, 9 vs 33 y 10 vs 34) ocurre lo contrario y en los casos restantes resulta indistinto.

Con respecto a las rutas complementarias, comparamos las configuraciones que difieren únicamente en el uso de dichas rutas (configuraciones 1 vs 2, 3 vs 4, etc). Los resultados muestran que en 3 de estos casos la solución mejora si se utilizan rutas complementarias y no mejora en caso contrario (configuraciones 11 vs 12, 23 vs 24 y 41 vs 42) y en otros tres casos ocurre lo contrario (configuraciones 5 vs 5, 13 vs 14 y 19 vs 20). En el resto de los casos la utilización de las rutas complementarias no influyó en la mejora de la solución.

Instancia 2

La instancia 2 consiste de 6 ciudades, 6 pedidos, 4 camiones y 11 camioneros. La tabla A.8 muestra los resultados para las 48 corridas correspondientes a las diferentes configuraciones de parámetros. En esta instancia se obtuvieron mejoras de la solución inicial en 13 de las 48 configuraciones.

La mayoría de los casos de mejora de la solución inicial se dieron en configuraciones con generación de rutas muy restringida (configuraciones 17 y 41 a 46). Los 6 casos de mejoras restantes se distribuyen equitativamente entre configuraciones correspondientes a generación poco restringida e irrestricta. La mejor solución, con valor 78, se dio en configuraciones con generación muy restringida de rutas.

Con respecto a la cota superior para dominancia relajada, en 9 de los 13 casos de mejora dicho parámetro es infinito (configuraciones 1, 9, 17, 25, 26, 33, 34, 41 y 42). Dentro de éstos está también la mejor solución encontrada.

En 7 casos de mejora no se utilizaron las rutas potencialmente beneficiosas y en los 6 restantes sí. Los casos en los que se obtuvo la mejor solución corresponden a configuraciones que utilizan rutas potencialmente beneficiosas. Si realizamos la comparación de los pares de configuraciones que difieren en el uso de estas rutas, tenemos que en 4 casos (configuraciones 19 vs 43, 20 vs 44, 21 vs 45 y 22 vs 46) la mejora se da cuando no se utilizan, en 3 casos (configuraciones 2 vs 26, 10 vs 34 y 18 vs 42) cuando sí se utilizan y en los casos restantes resulta indistinto.

En cuanto a las rutas complementarias, observamos que en 3 casos se obtienen mejores resultados sin utilizar dichas rutas con respecto a la misma configuración que sí las utiliza (configuraciones 1 vs 2, 9 vs 10 y 17 vs 18). En el resto el hecho de utilizar rutas complementarias fue indistinto.

Instancia 3

La instancia 3 consiste de 6 ciudades, 8 pedidos, 6 camiones y 12 camioneros. Los resultados obtenidos para las 48 combinaciones de parámetros para esta instancia pueden encontrarse en la tabla A.12. En esta instancia, con 11 de las 48 combinaciones de parámetros utilizadas se obtuvieron soluciones de menor costo que la solución inicial.

De las 11 configuraciones de parámetros que permitieron mejorar la solución inicial, 9 de ellas corresponden a configuraciones que restringen la generación de rutas. En 6 de dichas configuraciones (configuraciones 24, 43 a 46 y 48), la generación de rutas fue muy restringida, y en las restantes (13, 33 y 34) fue medianamente restringida. La mejor solución se obtuvo en una configuración en la cual la generación de rutas estaba fuertemente restringida (configuración 45). Si bien con 2 configuraciones que no restringen la generación

de rutas se logró mejorar la solución inicial, la mejora en esos casos fue notablemente menor que en el resto de los casos.

En 4 de los 11 casos de mejora la cota superior para dominancia relajada era infinito. Los demás casos de mejora se distribuyen casi uniformemente entre los restantes valores posibles para dicha cota (2 casos con cota 5, 3 casos con cota 3, 2 casos con cota 1).

Con respecto a las rutas potencialmente beneficiosas, en 7 de las configuraciones de parámetros que brindaron una solución mejor que la inicial dichas rutas no se utilizaban, y en las otras 4 las rutas potencialmente beneficiosas sí eran tenidas en cuenta. En la comparación entre pares de configuraciones que difieren únicamente en el uso de las rutas potencialmente beneficiosas, tenemos que en 6 casos (configuraciones 1 vs 25, 2 vs 26, 19 vs 43, 20 vs 44, 21 vs 45 y 22 vs 46) la mejora se da cuando no se utilizan, en 4 casos (configuraciones 13 vs 37, 24 vs 48, 9 vs 33 y 10 vs 34) la mejora se da cuando sí se utilizan y en el resto resulta indistinto.

En cuanto a las rutas complementarias, en 4 casos se mejoró la solución inicial tanto si se agregaban rutas complementarias como si no. Es decir, 8 de los casos de mejora corresponden a 4 pares de configuraciones que difieren únicamente en el uso de rutas complementarias.

Instancia 4

La instancia 4 consiste de 6 ciudades, 10 pedidos, 6 camiones y 12 camioneros. La tabla A.16 muestra los resultados obtenidos para esta instancia con cada una de las 48 combinaciones de parámetros. Para esta instancia, sólo con 4 de las 48 combinaciones de parámetros se logró mejorar la solución inicial.

En todos los casos en los que la solución inicial se mejoró, la generación de rutas estaba restringida. En 3 de esos casos (configuraciones 17, 18 y 22) la generación de rutas estaba fuertemente restringida. El otro caso correspondía a una configuración en la cual la generación de rutas estaba medianamente restringida (configuración 35), y a pesar de que la solución en este caso es mejor que la solución inicial, resultó ser la que menos mejoró de las 4.

En 2 de los 4 casos en los que se mejoró la solución inicial, la cota para el rango de dominancia relajada era infinito. De los otros 2 casos de mejora, en uno la cota era 5 y en el otro 3.

Con respecto al uso de rutas potencialmente beneficiosas, en 3 casos de mejora no se utilizaron y en el caso restante sí. En la comparación entre pares de configuraciones que difieren en el uso de este parámetro, tenemos que en 3 casos (configuraciones 11 vs 35, 17 vs 41 y 18 vs 42) la mejora se da cuando no se utilizan rutas potencialmente beneficiosas, en un caso (configuraciones 22 vs 46) ocurre lo contrario y en el resto de los casos resulta indistinto.

Con respecto a las rutas complementarias, 2 de los casos en los que se mejoró la solución corresponden a una configuración idéntica de los parámetros, salvo porque en uno de ellos se utilizan rutas complementarias (configuración 17) y en el otro no (configuración 18). En otro de los casos, agregar rutas complementarias mejora la solución con respecto a no agregarlas. En el último caso de mejora no se utilizan rutas complementarias, y para igual configuración de los demás parámetros, pero haciendo uso de rutas complementarias, la solución no mejora. Esto indica que en ese caso particular las rutas complementarias empeoraban el rendimiento del algoritmo.

Instancia 5

La instancia 5 consiste de 6 ciudades, 10 pedidos, 7 camiones y 13 camioneros. La tabla A.20 muestra los resultados obtenidos para esta instancia con cada una de las 48 combinaciones de parámetros. En esta instancia se logró mejorar la solución inicial en 8 configuraciones.

Se obtuvieron mejoras en la solución inicial tanto en casos con rutas muy restringidas (configuraciones 23 y 24), como medianamente restringidas (configuraciones 11 a 14) y nada restringidas (configuraciones 4 y 6). La mejor solución obtenida, con valor 190, se encuentra en los casos de configuraciones con rutas muy restringidas.

No hay ningún caso de mejora en el que la cota para dominancia relajada sea infinito. En esta instancia pareciera que los valores de 5 y 3 para la cota funcionan de la misma forma, ya que para grupos de configuraciones similares (configuraciones 9 a 16) si la cota tiene un valor de 5 o de 3, se logra mejorar la solución inicial y, si toma otro valor, no sucede esto.

En todos los casos de mejora fueron utilizadas las rutas potencialmente beneficiosas, por lo tanto, en la comparación de pares de configuraciones que sólo difieren en este parámetro y que resultaron en una mejora (configuraciones 4 vs 28, 6 vs 30, 11 vs 35, 12 vs 36, 13 vs 37, 14 vs 38, 23 vs 47 y 24 vs 48) observamos que es beneficioso utilizar estas rutas frente a no utilizarlas.

En 3 pares de casos con similar configuración en donde se logró mejorar la solución inicial (configuraciones 11 vs 12, 13 vs 14 y 23 vs 24) el hecho de utilizar rutas complementarias fue irrelevante para la mejora: las soluciones alcanzadas son iguales tanto si se utilizan como si no. En los otros 2 casos (configuraciones 4 y 6), las rutas complementarias representan un beneficio con respecto a la no utilización de las mismas.

Instancia 6

La instancia 6 consiste de 7 ciudades, 12 pedidos, 8 camiones y 15 camioneros. En la tabla A.24 se muestran los resultados obtenidos para esta instancia con cada una de las 48 combinaciones de parámetros. Se logró mejorar la solución inicial en 6 de las 48 configuraciones.

En todos los casos de mejora de la solución inicial, la generación de rutas fue restringida. En 5 de los 6 casos de mejora (configuraciones 17, 19, 22, 43 y 44) se restringió mucho la generación de rutas. La mejor solución, con valor 124, de nuevo se encuentra dentro del caso correspondiente a una configuración con generación muy restringida de rutas.

En cuanto al valor de la cota para dominancia relajada, en 4 de los 6 casos de mejora se tiene como valor al 5 (configuraciones 11, 19, 43 y 44). Uno de los casos tiene el valor 3 (configuración 22), y el otro tenía el valor infinito (configuración 17). La mejor solución fue obtenida con esta última configuración.

Hay 3 casos de mejora en los que se utilizaron rutas potencialmente beneficiosas y 3 casos en donde no. Si comparamos los pares de configuraciones que difieren únicamente en este parámetro se observa que en 2 casos (configuraciones 11 vs 35 y 22 vs 46) fue beneficioso utilizar la generación de estas rutas, dado que la solución mejoró modificando ese único parámetro. En otros 2 casos (configuraciones 17 vs 41 y 20 vs 44) ocurre lo contrario y en un caso (configuraciones 19 vs 43) resultó indistinto con respecto a la mejora, aunque la solución mejoró considerablemente más en el caso en que no se consideraron estas rutas.

Con respecto a las rutas complementarias, si comparamos los pares de configuraciones que difieren en el parámetro de uso de rutas complementarias y en donde hubo alguna mejora de la solución inicial, en 3 casos (configuraciones 11 vs 12, 17 vs 18 y 19 vs 20) la mejora se da cuando no se utilizan rutas complementarias. Sólo en un caso (configuración 21 vs 22) la mejora se da cuando sí se utilizan rutas complementarias y en el último caso (configuración 43 vs 44) resulta indistinto.

7.2.3. Conclusiones

En todas las instancias se obtuvieron buenos resultados al restringir la generación de rutas. En la mayoría, los mejores resultados se dieron en configuraciones con la mayor restricción, aunque también se obtuvieron mejoras para las configuraciones de restricción moderada. En todos los casos, la mejor solución se obtuvo con alguna de las configuraciones más restringidas. Estos resultados parecen indicar que el hecho de restringir las rutas a generar permite reducir el espacio de búsqueda del algoritmo de generación sin quitarle la posibilidad de que encuentre rutas con costo reducido negativo para mejorar la solución. Asimismo, la aplicación de estas restricciones sobre la generación de rutas tiene sentido desde el punto de vista práctico, dado que se obtienen rutas que se concentran más en la realización de pedidos que en el traslado de camioneros entre ciudades sin un fin específico.

En cuanto a la cota superior para dominancia relajada, se obtuvieron diversos resultados, aunque en la mayoría de las instancias el hecho de poner un valor para la cota mejoró la solución en una mayor cantidad de veces que cuando no se estableció un límite para la misma. Con respecto al valor que toma la cota, no pudimos observar que alguno de los valores utilizados sea claramente mejor que los otros. Esto significa que una cota más ajustada no necesariamente permite obtener mejores resultados que una menos ajustada. De esto podemos concluir que conviene fijarle un valor a la cota sin ajustarla demasiado, dado que a medida que se ajusta la cota se amplía el espacio de búsqueda del algoritmo de generación y, por ende, se perjudica su *performance*.

Para el parámetro que indica la utilización de rutas potencialmente beneficiosas no pudimos establecer ninguna conclusión determinante. En la mayoría de las instancias se obtuvieron resultados levemente mejores cuando no se consideraba el uso de estas rutas. Sin embargo, en la instancia 5 resulta evidente que los resultados son mejores al considerarlas.

Con respecto a las rutas complementarias, no pudimos observar que una opción sea claramente mejor que la otra. Hay configuraciones en las cuales el uso de las rutas complementarias mejoró la solución, y hay otras en las que se dió el caso contrario. En la mayoría, sin embargo, resultó indistinto el uso de este tipo de rutas, es decir, el hecho de utilizar o no las rutas complementarias no influyó en la mejora de la solución.

7.3. Pruebas finales

En esta sección presentaremos los resultados obtenidos para las pruebas sobre instancias de mayor tamaño que las utilizadas en las pruebas preliminares. Estas instancias cuentan con 10 ciudades y 16 pedidos, y resultaron ser las instancias de mayor tamaño que pudimos procesar en el tiempo estipulado (12 hs. por configuración). Realizamos estas pruebas en una PC con procesador Intel® Core™ i7 3.40 GHz y 16 GB de RAM.

7.3.1. Parámetros

Dado que las instancias consideradas en las pruebas finales son de mayor tamaño y requieren de mucho más tiempo de procesamiento, fue necesario reducir la cantidad de configuraciones de parámetros para realizar las corridas.

En base a las conclusiones obtenidas en las pruebas preliminares, decidimos utilizar las siguientes opciones de parámetros presentadas en la tabla 7.2.

Configuración	Restricciones rutas				
	MCCV	CVR	MPCD	Cota	RPB
1	1	NO	20 %	5	NO
2	1	NO	20 %	5	SI
3	1	NO	20 %	20	NO
4	1	NO	20 %	20	SI
5	2	NO	50 %	5	NO
6	2	NO	50 %	5	SI
7	2	NO	50 %	20	NO
8	2	NO	50 %	20	SI
9	4	NO	70 %	5	NO
10	4	NO	70 %	5	SI
11	4	NO	70 %	20	NO
12	4	NO	70 %	20	SI

Tabla 7.2: Combinaciones de parámetros para pruebas definitivas

Para la categoría de restricciones sobre la generación de rutas decidimos utilizar tres alternativas de parámetros:

- Muy restringida: MMCV: 1, CVR: no, MPCD: 20 %
- Medianamente restringida: MCCV: 2, CVR: no, MPCD: 50 %
- Poco restringida: MMCV: 4, CVR: no, MPCD: 70 %

Dado que según los resultados de las pruebas preliminares pudimos concluir que el hecho de restringir la generación de rutas daba buenos resultados, decidimos tener tres alternativas de restricción y combinarlas con todo el resto de los parámetros.

En cuanto al parámetro de cota superior para dominancia relajada, descartamos utilizar la opción de infinito ya que observamos que en las pruebas preliminares no se obtuvieron buenos resultados. Decidimos combinar dos valores, 5 y 20, con el resto de los parámetros de forma de tener una opción de cota ajustada y otra menos ajustada según el tamaño de las instancias.

Con respecto a la utilización de rutas potencialmente beneficiosas, debido a que en algunos casos resultó mejor utilizarlas frente a no hacerlo y en otros se dio lo contrario, decidimos considerar ambas alternativas y combinarlas con todo el resto de los parámetros.

Sobre las rutas complementarias observamos que en la mayoría de los casos su utilización no influía en la mejora de las soluciones por lo que, en pos de reducir la cantidad de configuraciones, decidimos descartar la utilización de las mismas para estas pruebas.

7.3.2. Resultados computacionales

A continuación presentamos los resultados de las corridas sobre las instancias de prueba finales. Estas pruebas fueron realizadas limitando el tiempo de la etapa de generación de columnas a 8 horas, de la etapa de resolución a 4 horas y la cantidad de iteraciones del algoritmo de resolución general a 200 para todas las instancias. Las descripciones de cada instancia, así como las tablas con los datos resultantes de las corridas pueden encontrarse en el apéndice A.

Instancia 7

La instancia 7 consiste de 10 ciudades, 16 pedidos, 10 camiones y 18 camioneros. En la tabla A.28 se muestran los resultados obtenidos para cada una de las 12 configuraciones de parámetros.

En esta instancia se logró mejorar la solución inicial en la configuración número 10. En esta configuración se hace uso de las rutas potencialmente beneficiosas logrando mejorar la solución habiendo realizado una sola iteración de la heurística.

En muchas de las configuraciones (6 de 12) no se llegó a obtener ninguna ruta con costo reducido negativo. Esto no se observa en el resto de las instancias, en donde siempre se pudieron generar al menos dos rutas. Esto nos lleva a pensar que la razón por la cual esto sucede no radica en el algoritmo sino en alguna característica particular de esta instancia.

La excepción a la poca generación de rutas se da en la configuración que logró mejorar la solución inicial y en las configuraciones 1 a 4, que son aquellas con mayor restricción en la generación. En estos casos se alcanzaron a generar más de 20.000 rutas. Asimismo, en las configuraciones 1, 3 y 4 se obtuvo un valor positivo para el *gap*. Este valor se computa como $|mse - msr| \,/\, |mse|$, donde mse es la mejor solución entera encontrada hasta el momento y msr es la mejor solución de la relajación lineal del problema. El *gap* varía entre 0 y 1, y sirve como medida de cuánto puede llegar a mejorar la solución entera con respecto a la mejor que se tiene hasta ahora, dado que la misma se encuentra acotada por la mejor solución de la relajación. En estas configuraciones, si la solución mejorara tanto como lo indica el *gap*, la solución obtenida sería mucho mejor que la de la configuración 10, que tiene un valor de 0 para el *gap*.

Instancia 8

La instancia 8 consiste de 10 ciudades, 16 pedidos, 8 camiones y 14 camioneros. En la tabla A.32 se muestran los resultados obtenidos para las 12 configuraciones de parámetros.

Se logró mejorar la solución inicial en una única configuración: la número 8. En este caso la mejora de la solución fue sustancial, pasó de un valor de $262,7$ a $167,6$. Al igual que en el caso anterior la mejora se dio en una configuración que hacía uso de las rutas potencialmente beneficiosas.

En esta instancia también observamos que hay varias configuraciones que quedaron con un *gap* positivo. En las cuatro configuraciones con mayor valor de *gap* (configuraciones 1, 2, 4 y 9), si la solución hubiera mejorado tanto como indica el *gap*, se hubiera obtenido una solución mejor que la obtenida con la configuración número 8. Esta configuración concluyó con un valor de 0 para el *gap*.

En las configuraciones 2, 4 y 10, que consideran el uso de rutas potencialmente beneficiosas, el hecho de que el valor del *gap* sea positivo puede deberse al gran número de rutas generadas. Sin embargo, en las configuraciones 1, 3 y 9, el número de rutas generadas es

considerablemente menor y, en algunos casos, el valor del *gap* resultó incluso mayor que en las otras.

Las primeras cuatro configuraciones, correspondientes a la mayor restricción sobre la generación de rutas, concluyeron la etapa de generación sin alcanzar el límite de tiempo. En tres de ellas (configuraciones 1, 3 y 4) se alcanzó el límite de las 200 iteraciones. Este resultado muestra cómo una fuerte restricción sobre la generación de rutas reduce considerablemente el espacio de búsqueda para el algoritmo de generación, contribuyendo a que el mismo obtenga una solución en un menor tiempo. Esto se observa particularmente en las configuraciones 1 y 3, en donde se llegaron a completar 26 y 30 iteraciones del algoritmo de generación exacto respectivamente.

Instancia 9

Esta instancia consiste de 10 ciudades, 16 pedidos, 9 camiones y 16 camioneros. En la tabla A.36 se pueden ver los resultados de las corridas para cada una de las 12 configuraciones de parámetros.

En esta instancia se logró mejorar la solución inicial en una única configuración: la número 6. Esta configuración hace uso de las rutas potencialmente beneficiosas al igual que los casos de mejora en las instancias anteriores. En muchos de los casos el valor del *gap* resultó positivo y, al igual que en el caso anterior, en algunos se generaron muchas rutas (configuraciones 2, 4, 8, 10 y 12) mientras que en otros no (configuraciones 1, 3 y 7). En todos los casos donde el valor del *gap* resultó positivo, si la solución hubiera mejorado tanto como indica el *gap*, entonces la solución óptima hubiese resultado mejor que la obtenida con la configuración número 6 que concluyó con un valor de 0 para el *gap*.

En las configuraciones donde la restricción de generación es más estricta se observa que en dos de los casos (configuraciones 1 y 3) la etapa de generación finalizó al alcanzarse el límite de las 200 iteraciones mientras que en las restantes (configuraciones 2 y 4), la etapa de generación finalizó realizando menos de 30 iteraciones. Esto puede deberse a que estas dos últimas configuraciones hacen uso de las rutas potencialmente beneficiosas, lo cual permite obtener una cantidad mucho mayor de rutas y produce que la resolución de la relajación lineal del problema en cada iteración demore más.

Instancias 10 y 11

La instancia 10 consiste de 10 ciudades, 16 pedidos, 8 camiones y 18 camioneros. La instancia 11 consiste de 10 ciudades, 16 pedidos, 9 camiones y 16 camioneros. En las tablas A.40 y A.44 se pueden ver los resultados de las corridas para cada una de las 12 configuraciones de parámetros de cada instancia respectivamente.

En ambas instancias se obtuvieron resultados muy similares. En ninguna de las dos se logró mejorar la solución inicial para ninguna de las configuraciones. En las configuraciones 2, 4, 6, 8, 10 y 12, que hacen uso de las rutas potencialmente beneficiosas, se generó una gran cantidad de rutas y el valor del *gap* resultó positivo para todas ellas. Esto también ocurrió las configuraciones 1 y 3 las cuales no tienen en cuenta las rutas potencialmente beneficiosas. En estos casos, ocurrió lo mismo que en las instancias anteriores: con sólo 200 rutas generadas, se obtiene un valor positivo para el *gap* no pudiendo mejorar la solución inicial.

7.3.3. Conclusiones

Todas las configuraciones que lograron mejorar la solución inicial contaron con un número importante de rutas generadas: 874 en la instancia 7, 5859 en la instancia 8 y 4636 en la instancia 9. Esto se debió a que dichas configuraciones hacían uso de las rutas potencialmente beneficiosas, por lo que en cada iteración, se generaron varias rutas de costo reducido negativo. De esto podemos concluir que es importante buscar una combinación de parámetros que permita que se generen suficientes rutas como para que en la etapa de resolución se tenga más chance de conseguir una solución factible mejor que la inicial.

En general, pareciera que a mayor cantidad de rutas, más tiempo se requiere para la etapa de resolución. Esto se puede observar en que, en la mayoría de los casos, a mayor cantidad de rutas se obtiene un valor más grande para el *gap*. Es importante aclarar que esto no sucede en todos los casos y, de hecho, hubo casos con pocas rutas generadas en donde el *gap* fue considerablemente grande. Esto ocurre, por ejemplo, en las configuraciones 1 y 3 de todas las instancias, en donde se alcanzaron a realizar las 200 iteraciones generando de esta manera un total de 200 rutas.

Las configuraciones con generación de rutas más restringidas son las que lograron producir el mayor número de rutas y, en aquellos casos en donde no se utilizaron las rutas potencialmente beneficiosas, se completaron las 200 iteraciones establecidas. Esto refuerza un poco la idea de que restringir fuertemente la generación de rutas agiliza cada iteración de la etapa de generación de columnas. De esta forma se pueden obtener una mayor cantidad de rutas lo que permite que en la etapa de resolución se cuenten con más alternativas para mejorar la solución inicial. Si bien creemos que el hecho de contar con más rutas generadas tiene un mayor potencial de mejorar la solución inicial, también produce que la etapa de resolución demore más, al punto de que ninguna de estas configuraciones logró finalizar con un valor de 0 para el *gap*.

Dada la cantidad de configuraciones que terminaron con un valor de *gap* positivo, podemos concluir que el límite de tiempo que establecimos para la etapa de resolución no fue suficiente. En relación a esto es importante rescatar que en base a lo mencionado anteriormente, el hecho de generar más rutas hace necesario aumentar el tiempo dedicado a la etapa de resolución. Es por esto que sería necesario realizar más pruebas para encontrar el equilibrio entre los tiempos de generación y resolución con el objetivo de encontrar soluciones factibles mejores a la inicial en un mayor número de casos.

Capítulo 8

Conclusiones

En este trabajo hemos propuesto un modelo de programación lineal entera binaria junto con un método de generación de columnas para resolver un problema combinado de ruteo de vehículos y *scheduling* de camioneros inspirado en un problema real. Tanto el problema de ruteo de vehículos como el de *scheduling* de conductores son difíciles de resolver computacionalmente, aún tratándolos en forma independiente. El objetivo de este trabajo fue resolver ambos problemas con un enfoque integrado, en contraposición a resolverlos de manera secuencial, lo cual aumenta considerablemente la dificultad.

El problema abordado conjuga varios aspectos de problemas conocidos como el *Vehicle and Crew Scheduling Problem* y el *Pick-up and Delivery Problem with Time Windows*. Sin embargo, no encontramos ningún problema en la literatura cuya definición agrupara las mismas características que el tratado en este trabajo. Como consecuencia de esto tuvimos que plantear un modelo de programación lineal original, el cual, dada la gran cantidad de características que tiene el problema, no pudo ser considerado en forma explícita con todas sus variables.

Al no considerar el modelo en su totalidad, decidimos utilizar la técnica de generación de columnas para generar aquellas variables que representan los posibles recorridos que puede realizar un vehículo. Para esto desarrollamos un modelo de generación basado en una red dirigida con recursos y restricciones que denominamos *grafo de rutas*. El objetivo de este modelo era llevar el problema de generación de rutas a un problema conocido de forma de contar con recursos teóricos para poder resolverlo. Como resultado, el problema de generar una columna de costo reducido negativo se tradujo a encontrar un camino de costo mínimo en el grafo de rutas.

Si bien el problema de camino mínimo sobre un grafo está bien resuelto, no pudimos utilizar los algoritmos conocidos debido a que nuestro grafo considera recursos y restricciones sobre los mismos. Por este motivo desarrollamos un algoritmo basado en la técnica de programación dinámica que nos permite obtener un camino de costo óptimo que verifique las restricciones sobre los recursos que considera el grafo de rutas. El problema que surgió al implementarlo fue que no era lo suficientemente eficiente en términos de tiempo de ejecución para utilizarlo en todas las iteraciones del algoritmo de resolución general. Con el objetivo de mejorar el tiempo de ejecución ideamos dos modificaciones que permiten reducir el espacio de búsqueda del algoritmo: restringir la estructura de las rutas a generar y relajar la relación de dominancia entre caminos. Ambas modificaciones cumplieron el objetivo permitiendo mejorar el tiempo de ejecución del algoritmo implementado.

Otro problema que presentó la generación de rutas fue que, en muchos casos, las columnas generadas mediante este método no conformaban una solución factible entera para el problema. Por este motivo implementamos la generación de dos tipos de rutas: poten-

cialmente beneficiosas y complementarias. Estas alternativas al mecanismo tradicional de generación de columnas son un aporte original de este trabajo y, a nuestro conocimiento, no tienen antecedente de haber sido implementadas. Como mencionamos en el capítulo 7, no pudimos obtener una conclusión sólida acerca de la utilización de estas rutas, aunque consideramos que fue interesante la implementación de estas ideas como alternativa a la generación de columnas tradicional.

Con respecto a los resultados obtenidos, resulta difícil juzgar la calidad de los mismos dado que, al tratarse de un problema sin antecedente en la literatura, no contamos con resultados previos con los cuales poder realizar una comparación. Las variantes implementadas sobre el algoritmo de generación de columnas constituyen un conjunto de parámetros que nos permite considerar una amplia gama de opciones al momento de resolver el problema. Como contrapartida, el estudio exhaustivo de las combinaciones de parámetros es muy costoso y, por lo tanto, resulta difícil determinar que combinaciones conducen a buenos resultados.

En todas las instancias relativamente chicas sobre las que se realizaron pruebas, la solución inicial pudo mejorarse con varias de las combinaciones de parámetros utilizadas. En las instancias de mayor tamaño también se logró mejorar la solución con algunas combinaciones de parámetros, aunque la mejora fue menor con respecto a las instancias más chicas. Como conclusión de los resultados obtenidos en las instancias de mayor tamaño consideramos que necesitamos incrementar el límite de tiempo de ejecución, en particular para la etapa de resolución, para ver si se logra mejorar la solución inicial en un mayor número de casos.

Trabajo a futuro

La necesidad de acotar el alcance del trabajo nos condujo a relegar ciertos aspectos que consideramos hubieran sido interesantes de explorar.

El algoritmo de resolución general propuesto en este trabajo contempla la generación de columnas únicamente en el nodo raíz del árbol de *Branch and Cut*. El hecho de que el algoritmo de generación no sea capaz de encontrar nuevas columnas de costo reducido negativo para este nodo, no permite garantizar que en el resto de los nodos del árbol no se den las condiciones para que esto sí suceda. Este método es conocido como *Branch and Price* e implicaría que el método de generación de columnas sea capaz de considerar las restricciones locales de *branching* para cada nodo. Sería interesante analizar qué impacto tienen estas restricciones en el método de generación que desarrollamos y, de ser posible, implementarlo.

Otro aspecto sobre el que sería atractivo profundizar sería considerar optimizaciones para el algoritmo de camino óptimo utilizado en la etapa de generación de columnas. De la misma forma que redefinimos la relación de dominancia con el objetivo de reducir el espacio de búsqueda del algoritmo, se podrían analizar otras posibles formas de relajación de esta relación de manera de reducir el tiempo de ejecución del mismo.

En este trabajo se hizo hincapié en implementar optimizaciones para el subproblema de generación de columnas. Sin embargo, no analizamos optimizaciones sobre el algoritmo de *Branch and Cut* utilizado para resolver el problema principal. Dado que en muchas de las pruebas que realizamos no se alcanzó el óptimo dadas las columnas consideradas, sería interesante analizar optimizaciones sobre esta etapa con el objetivo de mejorar estos resultados. Algunas de estas optimizaciones podrían ser la aplicación de cortes específicos o generales y la utilización de distintas estrategias de *branching*.

Otro aspecto que no estudiamos fue el impacto que tiene la estructura de una instancia del problema en el comportamiento del algoritmo de resolución general. Algunas características que se podrían considerar son: la distribución de las ciudades en el mapa, la ubicación inicial de los camiones y camioneros, y la distribución tanto temporal como espacial de los pedidos. Sería interesante analizar si existe alguna relación entre estas características estructurales de las instancias y la combinación de parámetros que mejor resultado obtiene para cada una de ellas.

Hasta donde llega nuestro conocimiento, la innovación de nuestro trabajo consiste en haber considerado, en forma conjunta, un problema de ruteo de vehículos y *scheduling* de conductores surgido de la práctica, que combina características de varios problemas ya conocidos. Si bien por el momento las instancias de prueba tratadas no son de un gran tamaño, confiamos en haber sembrado una semilla para el tratamiento de este problema con mayor profundidad.

Bibliografía

[Chv83] Vašek Chvátal. *Linear Programming*. W. H. Freeman and Company, 1983.

[Des86] Martin Desrochers. *La Fabrication d' Horaires de Travail pour les Conducteurs d' Autobus par une Méthode de Génération de Colonnes*. PhD thesis, Centre de recherche sur les Transports, Université de Montréal, Montréal, Canada, 1986.

[Dij59] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[ENS08] Faramroze G. Engineer, George L. Nemhauser, and Martin W. P. Savelsbergh. Shortest path based column generation on large networks with many resource constraints. 2008.

[FH96] Christian Friberg and Knut Haase. An exact algorithm for the vehicle and crew scheduling problem. 1996.

[FHW03] R. Freling, D. Huisman, and A.P.M Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. 2003.

[Flo62] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.

[Ge08] Bruce L. Golden and S. Raghavan editors. *The Vehicle Routing Problem: Latest Advances and New Challenges (Operations Research/Computer Science Interfaces Series)*. Springer, 2008.

[Gom58] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Society*, 64:275–278, 1958.

[GP79] Martin Grötschel and Manfred W. Padberg. On the Symmetric Travelling Salesman Problem I: Inequalities. *Mathematical Programming*, 16:265–280, 1979. 10.1007/BF01582116.

[Hui04] Dennis Huisman. *Integrated and Dynamic Vehicle and Crew Scheduling*. PhD thesis, Erasmus Universiteit Rotterdam, Rotterdam, Nederland, 2004.

[IBM10] IBM®. *User's Manual for CPLEX®*. IBM, 2010.

[ID04] S. Irnich and G. Desaulniers. Shortest Path Problems with Resource Constraints. Technical Report G-2004-11, Les Cahiers du GERAD, HEC Montréal, Montréal, Quebec, Canada, 2004.

[Kar84] Narenda Karmankar. *A New Polynomial Time Algorithm for Lineal Programming*. *Combinatorica*, (4):373–395, 1984.

[MP08] Marta Mesquita and Ana Paias. Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. *Comput. Oper. Res.*, 35(5):1562–1575, May 2008.

[Pad73] M. W. Padberg. On the Facial Structure of Set Packing Polyhedra. *Mathematical Programming*, 5:199–215, 1973.

[Pap81] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, October 1981.

[Ste07] Ingmar Steinzen. *Topics in Integrated Vehicle and Crew Scheduling in Public Transport*. PhD thesis, Fakultät für Wirtschaftswissenschaften der Universität Paderborn, Paderborn, Deutschland, 2007.

[TV02] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. SIAM, Monographs on Discrete Mathematics and Applications, 2002.

[Wol98] Laurence A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., 1998.

Apéndice A

Instancias de pruebas

A.1. Instancia 1

A.1.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Milan	1	2
Verona	0	1
Bologna	2	3
Genova	1	2
Venecia	1	3
Florencia	2	2

Tabla A.1: Cantidad de camiones y camioneros por ciudad en el momento inicial

	Milan	Verona	Bologna	Genova	Venecia	Florencia
Milan	0.0	4.5	6.0	5.25	7.5	9.0
Verona	4.5	0.0	4.0	9.0	3.5	7.5
Bologna	6.0	4.0	0.0	9.0	4.5	4.5
Genova	5.25	9.0	9.0	0.0	10.5	7.5
Venecia	7.5	3.5	4.5	10.5	0.0	7.5
Florencia	9.0	7.5	4.5	7.5	7.5	0.0

Tabla A.2: Tiempo de viaje entre ciudades (en horas)

- Horizonte de planificación: $T = 6$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 4$
- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Milan	0	[2, 4]	Verona	0	[5, 7]	10
2	Verona	1	[6, 9]	Genova	1	[10, 13]	13.5
3	Bologna	2	[1, 3]	Venecia	2	[19, 21]	8
4	Genova	2	[10, 12]	Florencia	3	[15, 17]	9
5	Florencia	0	[4, 8]	Venecia	1	[10, 13]	7
6	Genova	1	[3, 7]	Milan	1	[16, 18]	15
7	Bologna	2	[18, 20]	Verona	3	[9, 11]	11.5
8	Venecia	1	[4, 8]	Verona	1	[10, 13]	9.5
9	Florencia	0	[3, 7]	Milan	0	[16, 18]	12
10	Genova	2	[5, 8]	Verona	2	[14, 20]	7.5

Tabla A.3: Pedidos

A.1.2. Resultados

Configuración	Solución			Tiempos		# Iteraciones		# Rutas		
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	RC	Total
1	167	0	167	601	6	7	1	0	0	6
2	167	0	167	601	7	7	1	0	0	6
3	167	0	167	601	7	6	1	455	0	460
4	167	0	167	601	6	5	0	283	64	351
5	127	0	167	601	25	5	0	1524	0	1528
6	167	0	167	602	7	2	0	426	70	497
7	167	0	167	601	2	2	0	893	0	894
8	167	0	167	604	3	2	0	893	8	902
9	167	0	167	601	7	8	1	0	0	7
10	167	0	167	601	7	8	1	0	0	7
11	167	0,05	167	601	300	6	1	1147	0	1152
12	132	0	167	601	36	3	0	567	166	735
13	159,5	0	167	601	15	5	0	1351	0	1356
14	167	0	167	602	3	1	0	652	110	763
15	167	0	167	601	3	1	0	1330	0	1331
16	167	0	167	605	3	1	0	1330	8	1339
17	**107**	0	167	75	13	20	11	0	0	31
18	**107**	0	167	73	12	20	11	0	0	31
19	**107**	0	167	74	99	14	5	1291	0	1310
20	167	0,36	167	464	300	11	4	1369	676	2060
21	**107**	0	167	159	8	18	3	1423	0	1444
22	**107**	0	167	424	97	16	5	1003	518	1542
23	167	0,36	167	580	300	20	1	4531	0	4552
24	112	0	167	610	21	2	0	646	295	943
25	167	0	167	601	11	4	1	197	0	202
26	167	0	167	601	9	4	1	197	0	202
27	167	0,03	167	601	300	30	0	0	0	30
28	167	0	167	601	46	8	0	0	2	10
29	167	0	167	601	7	4	1	0	0	5
30	167	0	167	601	7	4	1	0	2	7
31	167	0	167	601	7	1	0	0	0	1
32	167	0	167	601	8	1	0	0	0	1
33	132	0	167	601	20	10	1	259	0	270
34	132	0	167	601	19	13	1	254	4	272
35	157	0	167	601	15	5	1	0	0	6
36	157	0	167	601	20	5	1	0	2	8
37	163	0	167	601	20	7	0	0	0	7
38	163	0	167	601	17	7	0	0	3	10
39	167	0	167	601	7	1	0	0	0	1
40	167	0	167	601	7	1	0	0	0	1
41	167	0,36	167	109	300	12	4	1418	0	1434
42	**107**	0	167	118	41	19	5	1431	2	1457
43	**107**	0	167	56	23	22	7	0	0	29
44	**107**	0	167	59	7	24	5	0	5	34
45	**107**	0	167	25	83	17	4	0	0	21
46	**107**	0	167	33	18	19	6	0	4	29
47	**107**	0	167	32	11	18	0	0	0	18
48	**107**	0	167	60	101	30	1	0	3	34

Tabla A.4: Instancia 1 - resultados

A.2. Instancia 2

A.2.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Paris	1	2
Barcelona	0	2
Malaga	1	3
Madrid	1	3
Lisboa	0	0
Lyon	1	1

Tabla A.5: Cantidad de camiones y camioneros por ciudad en el momento inicial

	Paris	Barcelona	Malaga	Madrid	Lisboa	Lyon
Paris	0.0	4.0	7.0	8.0	11.0	2.0
Barcelona	4.0	0.0	3.0	4.0	5.0	3.0
Malaga	7.0	3.0	0.0	5.0	4.0	6.0
Madrid	8.0	4.0	5.0	0.0	3.0	7.0
Lisboa	11.0	5.0	4.0	3.0	0.0	10.0
Lyon	2.0	3.0	6.0	7.0	10.0	0.0

Tabla A.6: Tiempo de viaje entre ciudades (en horas)

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Malaga	0	[0, 23]	Madrid	0	[0, 23]	15
2	Madrid	0	[0, 23]	Barcelona	0	[0, 23]	26
3	Malaga	0	[0, 23]	Paris	0	[0, 23]	8
4	Lisboa	0	[0, 23]	Lyon	0	[0, 23]	14
5	Lyon	0	[0, 23]	Paris	0	[0, 23]	3.1
6	Lisboa	0	[0, 23]	Malaga	0	[0, 23]	11.3

Tabla A.7: Pedidos

- Horizonte de planificación: $T = 10$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 4$
- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

A.2.2. Resultados

Configuración	Solución			Tiempos		# Iteraciones		# Rutas		
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	RC	Total
1	125,3	0	145,3	601	62	4	0	0	0	4
2	145,3	0	145,3	601	27	2	0	0	2	4
3	145,3	0	145,3	601	22	0	0	0	0	0
4	145,3	0	145,3	601	22	0	0	0	0	0
5	145,3	0	145,3	601	23	0	0	0	0	0
6	145,3	0	145,3	601	22	0	0	0	0	0
7	145,3	0	145,3	601	22	0	0	0	0	0
8	145,3	0	145,3	601	22	0	0	0	0	0
9	125,3	0	145,3	601	50	4	0	0	0	4
10	144,2	0	145,3	1880	100	5	0	0	3	8
11	145,3	0	145,3	601	22	0	0	0	0	0
12	145,3	0	145,3	601	22	0	0	0	0	0
13	145,3	0	145,3	601	21	0	0	0	0	0
14	145,3	0	145,3	601	22	0	0	0	0	0
15	145,3	0	145,3	601	23	0	0	0	0	0
16	145,3	0	145,3	601	23	0	0	0	0	0
17	95	0,09	145,3	143	300	30	1	0	0	31
18	145,3	0,46	145,3	231	301	33	2	0	2	37
19	145,3	0,27	145,3	601	300	1	0	728	0	729
20	145,3	0,27	145,3	631	301	1	0	728	51	780
21	145,3	0,28	145,3	601	300	1	0	1164	0	1165
22	145,3	0,27	145,3	645	301	1	0	1164	11	1176
23	145,3	0	145,3	601	21	0	0	0	0	0
24	145,3	0	145,3	601	21	0	0	0	0	0
25	140,3	0	145,3	601	19	1	0	51	0	52
26	140,3	0	145,3	601	21	1	0	51	44	96
27	145,3	0	145,3	601	19	0	0	0	0	0
28	145,3	0	145,3	601	20	0	0	0	0	0
29	145,3	0	145,3	601	20	0	0	0	0	0
30	145,3	0	145,3	601	19	0	0	0	0	0
31	145,3	0	145,3	601	19	0	0	0	0	0
32	145,3	0	145,3	601	19	0	0	0	0	0
33	96	0	145,3	601	53	2	0	121	0	123
34	140,3	0	145,3	601	21	1	0	54	46	101
35	145,3	0	145,3	601	20	1	0	0	0	1
36	145,3	0	145,3	601	20	1	0	0	0	1
37	145,3	0	145,3	601	18	0	0	0	0	0
38	145,3	0	145,3	601	18	0	0	0	0	0
39	145,3	0	145,3	601	19	0	0	0	0	0
40	145,3	0	145,3	601	18	0	0	0	0	0
41	**78**	0	145,3	79	61	5	1	313	0	319
42	**78**	0	145,3	601	121	14	0	1757	511	2282
43	124	0,37	145,3	469	300	36	3	0	0	39
44	142	0,09	145,3	601	300	12	0	0	1	13
45	124,3	0	145,3	601	27	26	1	0	0	27
46	97,1	0	145,3	601	23	11	0	0	2	13
47	145,3	0	145,3	601	19	0	0	0	0	0
48	145,3	0	145,3	601	19	0	0	0	0	0

Tabla A.8: Instancia 2 - resultados

A.3. Instancia 3

A.3.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Tucson	1	2
Phoenix	0	2
Los Angeles	1	3
Las Vegas	1	1
San Diego	2	2
Albuquerque	1	2

Tabla A.9: Cantidad de camiones y camioneros por ciudad en el momento inicial

	Tucson	Phoenix	Los Angeles	Las Vegas	San Diego	Albuquerque
Tucson	0.0	1.75	7.75	6.8	6.3	7.0
Phoenix	1.75	0.0	6.0	5.3	5.9	7.0
Los Angeles	7.75	6.0	0.0	4.25	2.0	12.0
Las Vegas	6.8	5.3	4.25	0.0	5.45	9.0
San Diego	6.3	5.9	2.0	5.45	0.0	12.0
Albuquerque	7.0	7.0	12.0	9.0	12.0	0.0

Tabla A.10: Tiempo de viaje entre ciudades (en horas)

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Tucson	1	[0, 7]	Phoenix	2	[3, 8]	15
2	San Diego	1	[2, 5]	Las Vegas	1	[9, 12]	26
3	Los Angeles	0	[10, 14]	Las Vegas	0	[9, 18]	8
4	Las Vegas	0	[0, 23]	Albuquerque	0	[0, 23]	14
5	Albuquerque	0	[5, 13]	Tucson	0	[14, 18]	3.1
6	San Diego	0	[13, 14]	Albuquerque	1	[9, 12]	11.8
7	Phoenix	0	[0, 5]	Albuquerque	1	[19, 23]	7.9
8	Los Angeles	0	[18, 23]	Phoenix	0	[15, 19]	12.1

Tabla A.11: Pedidos

- Horizonte de planificación: $T = 8$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 4$
- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

A.3.2. Resultados

Configuración	Solución			Tiempos		# Iteraciones		# Rutas		
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	RC	Total
1	190,7	0	215,8	601	41	7	0	0	0	7
2	190,7	0	215,8	601	42	6	0	0	1	7
3	215,8	0,34	215,8	601	300	3	0	1226	0	1229
4	215,8	0	215,8	616	20	1	0	307	49	357
5	215,8	0	215,8	601	13	1	0	561	0	562
6	215,8	0	215,8	604	12	1	0	561	21	583
7	215,8	0	215,8	601	15	0	0	0	0	0
8	215,8	0	215,8	601	16	0	0	0	0	0
9	215,8	0,49	215,8	601	300	25	0	0	0	25
10	215,8	0,01	215,8	601	300	9	0	0	1	10
11	215,8	0	215,8	601	23	1	0	501	0	502
12	215,8	0	215,8	623	117	1	0	501	40	542
13	165,7	0	215,8	601	285	2	0	921	0	923
14	215,8	0	215,8	605	12	1	0	566	18	585
15	215,8	0	215,8	601	15	0	0	0	0	0
16	215,8	0	215,8	601	15	0	0	0	0	0
17	215,8	0,42	215,8	132	301	45	5	0	0	50
18	215,8	0,51	215,8	112	301	46	4	0	4	54
19	215,8	0,41	215,8	641	300	12	0	5084	0	5096
20	215,8	0,32	215,8	610	302	6	0	785	422	1213
21	215,8	0,32	215,8	610	301	6	0	4466	0	4472
22	215,8	0,26	215,8	643	300	2	0	1394	177	1573
23	215,8	0,54	215,8	593	301	16	1	3706	0	3723
24	159,7	0	215,8	615	53	1	0	358	57	416
25	215,8	0,34	215,8	601	300	4	0	239	0	243
26	215,8	0	215,8	606	18	2	0	244	71	317
27	215,8	0	215,8	601	15	5	0	0	0	5
28	215,8	0	215,8	601	14	5	0	0	1	6
29	215,8	0	215,8	601	12	2	0	0	0	2
30	215,8	0	215,8	601	13	2	0	0	0	2
31	215,8	0	215,8	601	15	0	0	0	0	0
32	215,8	0	215,8	601	15	0	0	0	0	0
33	141,1	0	215,8	601	147	6	0	180	0	186
34	144,7	0	215,8	601	15	4	0	35	3	42
35	215,8	0	215,8	601	15	4	0	0	0	4
36	215,8	0	215,8	601	15	4	0	0	0	4
37	215,8	0	215,8	601	15	3	0	0	0	3
38	215,8	0	215,8	601	15	3	0	0	0	3
39	215,8	0	215,8	601	15	0	0	0	0	0
40	215,8	0	215,8	601	15	0	0	0	0	0
41	215,8	0,52	215,8	612	300	39	5	2079	0	2123
42	215,8	0,46	215,8	603	300	20	1	2119	313	2453
43	124,6	0,17	215,8	320	301	42	8	0	0	50
44	107,1	0	215,8	368	60	46	4	0	17	67
45	**105**	0	215,8	270	23	36	6	0	0	42
46	107,1	0	215,8	232	19	47	3	0	8	58
47	215,8	0,26	215,8	601	300	13	0	0	0	13
48	116	0	215,8	601	115	45	3	0	9	57

Tabla A.12: Instancia 3 - resultados

A.4. Instancia 4

A.4.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Atlanta	2	2
Cincinnati	1	2
Chicago	0	1
Washington	1	2
Cleveland	1	2
Indianapolis	1	3

Tabla A.13: Cantidad de camiones y camioneros por ciudad en el momento inicial

	Atlanta	Cincinnati	Chicago	Washington	Cleveland	Indianapolis
Atlanta	0.0	6.4	7.5	10.5	11.8	8.75
Cincinnati	6.4	0.0	5.0	8.75	4.15	2.0
Chicago	7.5	5.0	0.0	12.0	5.5	3.25
Washington	10.5	8.75	12.0	0.0	6.5	9.7
Cleveland	11.8	4.15	5.5	6.5	0.0	5.0
Indianapolis	8.75	2.0	3.25	9.7	5.0	0.0

Tabla A.14: Tiempo de viaje entre ciudades (en horas)

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Indianapolis	0	[2, 8]	Washington	1	[9, 13]	3.6
2	Cincinnati	1	[8, 14]	Cleveland	2	[17, 22]	1.3
3	Atlanta	0	[13, 16]	Washington	2	[19, 22]	2.3
4	Chicago	0	[13, 14]	Cincinnati	2	[18, 20]	1.9
5	Chicago	0	[1, 9]	Cleveland	2	[11, 13]	7.1
6	Atlanta	0	[4, 8]	Cleveland	1	[12, 17]	3.4
7	Cleveland	0	[13, 18]	Cincinnati	0	[16, 22]	6
8	Atlanta	1	[2, 7]	Chicago	1	[14, 23]	4.2
9	Cincinnati	0	[14, 18]	Chicago	0	[12, 16]	3.9
10	Atlanta	0	[0, 4]	Indianapolis	0	[3, 8]	5

Tabla A.15: Pedidos

- Horizonte de planificación: $T = 8$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 1{,}6$
- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

A.4.2. Resultados

Configuración	Solución			Tiempos		# Iteraciones		# Rutas		
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	RC	Total
1	212,3	0	212,3	601	11	2	0	0	0	2
2	212,3	0	212,3	601	12	2	0	0	0	2
3	212,3	0	212,3	601	14	0	0	0	0	0
4	212,3	0	212,3	601	14	0	0	0	0	0
5	212,3	0	212,3	601	14	0	0	0	0	0
6	212,3	0	212,3	601	15	0	0	0	0	0
7	212,3	0	212,3	601	15	0	0	0	0	0
8	212,3	0	212,3	601	14	0	0	0	0	0
9	212,3	0	212,3	601	11	2	0	0	0	2
10	212,3	0	212,3	601	11	2	0	0	0	2
11	212,3	0,16	212,3	601	300	1	0	546	0	547
12	212,3	0,16	212,3	617	300	1	0	546	1	548
13	212,3	0	212,3	601	14	0	0	0	0	0
14	212,3	0	212,3	601	14	0	0	0	0	0
15	212,3	0	212,3	601	15	0	0	0	0	0
16	212,3	0	212,3	601	14	0	0	0	0	0
17	**165,1**	0	212,3	381	85	23	27	0	0	50
18	**165,1**	0	212,3	379	84	23	27	0	0	50
19	212,3	0,3	212,3	601	300	16	0	3888	0	3904
20	212,3	0,03	212,3	642	300	4	0	1634	540	2178
21	212,3	0,21	212,3	601	302	4	0	3254	0	3258
22	185,3	0	212,3	601	33	1	0	203	70	274
23	212,3	0	212,3	601	23	1	0	444	0	445
24	212,3	0	212,3	616	115	1	0	444	9	454
25	212,3	0	212,3	601	15	0	0	0	0	0
26	212,3	0	212,3	601	15	0	0	0	0	0
27	212,3	0	212,3	601	14	0	0	0	0	0
28	212,3	0	212,3	601	14	0	0	0	0	0
29	212,3	0,26	212,3	608	301	22	2	2191	0	2215
30	212,3	0,26	212,3	608	300	25	2	1968	35	2030
31	212,3	0,2	212,3	248	300	44	6	0	0	50
32	212,3	0,16	212,3	392	300	46	4	0	11	61
33	212,3	0,22	212,3	610	304	41	5	0	0	46
34	212,3	0,23	212,3	610	300	27	0	0	6	33
35	208,7	0,14	212,3	601	300	15	0	0	0	15
36	212,3	0,02	212,3	601	300	5	0	0	1	6
37	212,3	0	212,3	601	15	2	0	3	0	5
38	212,3	0	212,3	601	15	2	0	3	0	5
39	212,3	0	212,3	601	19	0	0	0	0	0
40	212,3	0	212,3	601	19	0	0	0	0	0
41	212,3	0	212,3	601	19	0	0	0	0	0
42	212,3	0	212,3	601	19	0	0	0	0	0
43	212,3	0	212,3	601	14	0	0	0	0	0
44	212,3	0	212,3	601	14	0	0	0	0	0
45	212,3	0	212,3	601	11	2	0	3	0	5
46	212,3	0	212,3	601	11	2	0	3	0	5
47	212,3	0	212,3	601	9	3	0	0	0	3
48	212,3	0	212,3	601	20	0	0	0	0	0

Tabla A.16: Instancia 4 - resultados

A.5. Instancia 5

A.5.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Madrid	2	3
Córdoba	1	2
Valencia	1	0
Barcelona	2	4
Bilbao	0	2
Zaragoza	1	2

Tabla A.17: Cantidad de camiones y camioneros por ciudad en el momento inicial

	Madrid	Córdoba	Valencia	Barcelona	Bilbao	Zaragoza
Madrid	0.0	4.25	3.8	6.0	4.0	3.45
Córdoba	4.25	0.0	5.7	8.3	8.0	7.2
Valencia	3.8	5.7	0.0	3.25	5.8	3.2
Barcelona	6.0	8.3	3.25	0.0	2.6	5.2
Bilbao	4.0	8.0	5.8	2.6	0.0	2.7
Zaragoza	3.45	7.2	3.2	5.2	2.7	0.0

Tabla A.18: Tiempo de viaje entre ciudades (en horas)

	Pick-up			*Drop-off*			
Pedido	Ciudad	Día	Ventana	Ciudad	Día	Ventana	Penalidad diaria
1	Madrid	0	[3, 7]	Bilbao	0	[8, 10]	10
2	Barcelona	1	[5, 8v	Zaragoza	2	[14, 16]	13.5
3	Bilbao	2	[1, 3]	Córdoba	2	[19, 21]	8
4	Valencia	2	[9, 11]	Zaragoza	3	[16, 21]	9
5	Madrid	0	[4, 8]	Barcelona	1	[10, 13]	7
6	Zaragoza	1	[4, 6]	Madrid	1	[10, 14]	15
7	Bilbao	2	[16, 18]	Córdoba	3	[9, 11]	11.5
8	Valencia	1	[6, 10]	Barcelona	1	[19, 21]	9.5
9	Madrid	0	[3, 7]	Córdoba	0	[16, 18]	12
10	Valencia	2	[7, 10]	Barcelona	2	[14, 20]	7.5

Tabla A.19: Pedidos

- Horizonte de planificación: $T = 6$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 6$
- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

A.5.2. Resultados

Configuración	Solución			Tiempos		# Iteraciones		# Rutas		
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	RC	Total
1	312	0	312	601	7	2	0	0	0	2
2	312	0	312	601	7	2	0	0	0	2
3	312	0,15	312	601	300	1	0	343	0	344
4	264	0	312	611	219	1	0	343	62	406
5	312	0,14	312	601	300	1	0	511	0	512
6	267	0	312	615	197	1	0	511	27	539
7	312	0	312	601	9	0	0	0	0	0
8	312	0	312	601	7	0	0	0	0	0
9	312	0	312	601	7	2	0	0	0	2
10	312	0	312	601	7	2	0	0	0	2
11	256	0	312	601	274	1	0	374	0	375
12	256	0	312	612	162	1	0	374	76	451
13	267	0	312	601	142	1	0	538	0	539
14	267	0	312	613	271	1	0	538	28	567
15	312	0,2	312	601	300	1	0	1477	0	1478
16	312	0,2	312	621	300	1	0	1477	0	1478
17	312	0,44	312	127	300	45	5	0	0	50
18	312	0,35	312	166	300	44	6	0	13	63
19	312	0,52	312	490	301	17	1	7480	0	7498
20	312	0,09	312	613	301	2	0	1652	271	1925
21	312	0,5	312	602	300	14	0	8603	0	8617
22	312	0,17	312	601	300	1	0	128	67	196
23	**190**	0	312	601	172	1	0	577	0	578
24	**190**	0	312	610	109	1	0	577	5	583
25	312	0,41	312	134	301	8	3	577	363	951
26	312	0,47	312	586	300	42	8	0	0	50
27	312	0,18	312	601	300	21	3	0	5	29
28	312	0,46	312	290	300	43	7	0	0	50
29	312	0,43	312	217	301	49	1	0	9	59
30	312	0	312	601	8	2	0	0	0	2
31	312	0	312	601	7	2	0	0	0	2
32	312	0	312	601	7	1	0	5	0	6
33	312	0	312	601	7	1	0	5	0	6
34	312	0	312	601	9	3	0	0	0	3
35	312	0	312	601	8	3	0	0	0	3
36	312	0	312	601	7	2	0	0	0	2
37	312	0	312	601	7	2	0	0	0	2
38	312	0	312	601	8	0	0	0	0	0
39	312	0	312	601	8	0	0	0	0	0
40	312	0	312	601	7	1	0	5	0	6
41	312	0	312	601	7	1	0	5	0	6
42	312	0	312	601	8	5	0	0	0	5
43	312	0	312	601	12	3	0	0	0	3
44	312	0	312	601	19	0	0	0	0	0
45	312	0	312	601	9	3	0	0	0	3
46	312	0	312	601	8	0	0	0	0	0
47	312	0	312	601	8	0	0	0	0	0
48	312	0,51	312	163	301	19	5	1004	0	1028

Tabla A.20: Instancia 5 - resultados

A.6. Instancia 6

A.6.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Berlin	2	3
Hannover	1	2
Hamburg	1	2
Koln	1	0
Frankfurt	2	4
Stuttgart	0	2
Munchen	1	2

Tabla A.21: Cantidad de camiones y camioneros por ciudad en el momento inicial

	Berlin	Hannover	Hamburg	Koln	Frankfurt	Stuttgart	Munchen
Berlin	0.0	2.75	2.8	5.4	5.1	6.0	5.75
Hannover	2.75	0.0	1.6	2.85	3.25	5.0	6.0
Hamburg	2.8	1.6	0.0	3.85	4.5	6.15	7.25
Koln	5.4	2.85	3.85	0.0	1.9	3.5	5.4
Frankfurt	5.1	3.25	4.5	1.9	0.0	2.0	3.75
Stuttgart	6.0	5.0	6.15	3.5	2.0	0.0	2.2
Munchen	5.75	6.0	7.25	5.4	3.75	2.2	0.0

Tabla A.22: Tiempo de viaje entre ciudades (en horas)

- Horizonte de planificación: $T = 6$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 6$
- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Berlin	0	[4, 8]	Hannover	0	[8, 10]	10
2	Hamburg	1	[7, 9]	Koln	2	[13, 16]	13.5
3	Munchen	2	[1, 3]	Frankfurt	2	[19, 21]	8
4	Koln	2	[15, 17]	Stuttgart	3	[16, 21]	9
5	Hannover	0	[4, 8]	Hamburg	1	[10, 13]	7
6	Stuttgart	1	[7, 10]	Berlin	1	[15, 18]	15
7	Hamburg	2	[16, 18]	Munchen	3	[9, 11]	11.5
8	Berlin	1	[6, 10]	Koln	1	[19, 21]	9.5
9	Frankfurt	0	[3, 7]	Hannover	0	[16, 18]	12
10	Munchen	2	[7, 10]	Berlin	2	[14, 20]	7.5
11	Hamburg	2	[5, 8]	Stuttgart	3	[21, 23]	11.5
12	Koln	2	[9, 13]	Berlin	3	[14, 18]	8

Tabla A.23: Pedidos

A.6.2. Resultados

Configuración	Solución			Tiempos		# Iteraciones		# Rutas		
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	RC	Total
1	270	0	270	601	11	4	0	0	0	4
2	270	0	270	601	11	4	0	0	2	6
3	270	0,12	270	601	300	2	0	1181	0	1183
4	270	0	270	605	18	1	0	752	32	785
5	270	0	270	601	19	1	0	763	0	764
6	270	0	270	605	19	1	0	763	4	768
7	270	0	270	601	12	0	0	0	0	0
8	270	0	270	601	12	0	0	0	0	0
9	270	0	270	601	10	7	0	0	0	7
10	270	0	270	601	10	7	0	0	4	11
11	248,5	0	270	601	50	2	0	1170	0	1172
12	270	0	270	605	14	1	0	907	49	957
13	270	0	270	601	19	1	0	845	0	846
14	270	0	270	605	17	1	0	845	10	856
15	270	0	270	601	12	0	0	0	0	0
16	270	0	270	601	14	0	0	0	0	0
17	**124**	0	270	220	12	36	14	0	0	50
18	270	0,49	270	319	301	46	4	0	8	58
19	242	0	270	601	109	1	0	2212	0	2213
20	270	0,1	270	623	300	1	0	2212	24	2237
21	270	0,07	270	601	301	2	0	3633	0	3635
22	252	0	270	624	250	1	0	2862	22	2885
23	270	0	270	601	13	0	0	0	0	0
24	270	0	270	601	13	0	0	0	0	0
25	270	0	270	601	12	2	0	84	0	86
26	270	0	270	601	11	2	0	86	88	176
27	270	0	270	601	13	1	0	0	0	1
28	270	0	270	601	13	1	0	0	0	1
29	270	0	270	601	14	0	0	0	0	0
30	270	0	270	601	14	0	0	0	0	0
31	270	0	270	601	14	0	0	0	0	0
32	270	0	270	601	14	0	0	0	0	0
33	270	0	270	601	12	2	0	69	0	71
34	270	0	270	601	12	2	0	69	66	137
35	270	0	270	601	13	1	0	0	0	1
36	270	0	270	601	13	1	0	0	0	1
37	270	0	270	601	13	1	0	0	0	1
38	270	0	270	601	12	1	0	0	0	1
39	270	0	270	601	14	0	0	0	0	0
40	270	0	270	601	14	0	0	0	0	0
41	270	0,54	270	73	300	13	4	737	0	754
42	270	0,54	270	91	300	11	3	490	132	636
43	126	0	270	570	17	38	12	0	0	50
44	252	0	270	601	50	15	4	0	4	23
45	270	0	270	601	17	9	0	0	0	9
46	270	0	270	601	17	6	0	0	2	8
47	270	0	270	601	14	0	0	0	0	0
48	270	0	270	601	15	0	0	0	0	0

Tabla A.24: Instancia 6 - resultados

A.7. Instancia 7

A.7.1. Definición de la instancia

- Horizonte de planificación: $T = 6$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 2{,}2$

Ciudad	Cant. camiones	Cant. camioneros
London	2	4
Brighton	0	1
Southampton	1	2
Bristol	0	1
Plymouth	1	2
Birmingham	1	1
Sheffield	1	2
Liverpool	1	1
Manchester	1	1
Leeds	2	3

Tabla A.25: Cantidad de camiones y camioneros por ciudad en el momento inicial

	London	Brighton	Southampton	Bristol	Plymouth	Birmingham	Sheffield	Liverpool	Manchester	Leeds
London	0.0	1.3	1.6	2.2	4.1	2.0	2.9	3.7	3.5	3.25
Brighton	1.3	0.0	1.5	2.75	4.4	2.8	3.75	4.5	4.3	4.2
Southampton	1.6	1.5	0.0	1.8	3.2	2.5	3.6	4.0	3.0	4.2
Bristol	2.2	2.75	1.8	0.0	2.25	1.5	2.9	3.0	2.9	3.4
Plymouth	4.1	4.4	3.2	2.25	0.0	3.5	4.9	4.9	4.75	5.4
Birmingham	2.0	2.8	2.5	1.5	3.5	0.0	1.5	1.6	1.5	2.0
Sheffield	2.9	3.75	3.6	2.9	4.9	1.5	0.0	1.5	1.0	0.7
Liverpool	3.7	4.5	4.0	3.0	4.9	1.6	1.5	0.0	0.6	1.25
Manchester	3.5	4.3	3.0	2.9	4.75	1.5	1.0	0.6	0.0	0.8
Leeds	3.25	4.2	4.2	3.4	5.4	2.0	0.7	1.25	0.8	0.0

Tabla A.26: Tiempo de viaje entre ciudades (en horas)

- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	London	0	[2, 4]	Brighton	0	[7, 13]	3.1
2	Southampton	0	[2, 6]	Bristol	1	[14, 18]	2.7
3	Liverpool	0	[13, 18]	Sheffield	0	[17, 20]	2.8
4	Leeds	0	[12, 14]	London	1	[3, 6]	1.5
5	Southampton	0	[13, 16]	Plymouth	2	[13, 16]	4
6	London	1	[2, 4]	Birmingham	1	[8, 14]	2.2
7	Liverpool	1	[17, 19]	Birmingham	1	[21, 23]	2.7
8	London	0	[4, 5]	Bristol	2	[12, 15]	4.9
9	Birmingham	0	[19, 21]	Sheffield	1	[22, 23]	2.3
10	Manchester	2	[12, 14]	Liverpool	3	[20, 23]	2.8
11	Sheffield	1	[2, 4]	London	3	[13, 17]	1.6
12	Leeds	2	[18, 20]	Liverpool	2	[14, 17]	1.8
13	Southampton	0	[13, 16]	Brighton	1	[15, 21]	5.3
14	London	1	[13, 15]	Leeds	2	[6, 9]	3.1
15	Sheffield	0	[4, 9]	Liverpool	0	[3, 6]	2
16	Plymouth	3	[19, 21]	Southampton	4	[22, 23]	0.8

Tabla A.27: Pedidos

A.7.2. Resultados

Configuración	Solución			Tiempos (seg.)		# Iteraciones		# Rutas	
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	Total
1	195,2	0,587602	195,2	23446	14402	167	33	0	200
2	195,2	0,409205	195,2	28801	14406	2	1	21550	21553
3	195,2	0	195,2	28802	224	12	4	0	16
4	195,2	0,572234	195,2	28003	14405	19	1	23927	23947
5	195,2	0	195,2	28802	1	0	0	0	0
6	195,2	0	195,2	28802	1	0	0	0	0
7	195,2	0	195,2	28802	1	0	0	0	0
8	195,2	0	195,2	28802	1	0	0	0	0
9	195,2	0	195,2	28801	2	4	0	0	4
10	**192**	0	195,2	28802	268	1	0	873	874
11	195,2	0	195,2	28802	1	0	0	0	0
12	195,2	0	195,2	28802	1	0	0	0	0

Tabla A.28: Instancia 7 - resultados

A.8. Instancia 8

A.8.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Paris	2	3
Reims	0	1
Nancy	1	2
Dijon	1	1
Orleans	0	1
Nantes	1	1
Bordeaux	1	1
Tolouse	0	1
Marseille	1	2
Lyon	1	1

Tabla A.29: Cantidad de camiones y camioneros por ciudad en el momento inicial

- Horizonte de planificación: $T = 5$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 2{,}2$
- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

	Paris	Reims	Nancy	Dijon	Orleans	Nantes	Bordeaux	Tolouse	Marseille	Lyon
Paris	0.0	1.7	3.8	3.1	1.5	3.9	5.9	6.3	7.3	4.25
Reims	1.7	0.0	2.5	3.0	2.7	5.0	7.0	7.5	7.8	4.75
Nancy	3.8	2.5	0.0	2.25	4.1	7.25	8.7	8.9	7.0	4.0
Dijon	3.1	3.0	2.25	0.0	3.0	6.0	6.8	6.6	4.8	1.75
Orleans	1.5	2.7	4.1	3.0	0.0	3.2	4.6	4.9	7.25	4.1
Nantes	3.9	5.0	7.25	6.0	3.2	0.0	3.7	5.75	9.4	6.75
Bordeaux	5.9	7.0	8.7	6.8	4.6	3.7	0.0	2.3	6.0	5.7
Tolouse	6.3	7.5	8.9	6.6	4.9	5.75	2.3	0.0	3.75	5.0
Marseille	7.3	7.8	7.0	4.8	7.25	9.4	6.0	3.75	0.0	3.1
Lyon	4.25	4.75	4.0	1.75	4.1	6.75	5.7	5.0	3.1	0.0

Tabla A.30: Tiempo de viaje entre ciudades (en horas)

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Paris	0	[2, 4]	Marseille	0	[7, 13]	3.1
2	Reims	0	[2, 6]	Dijon	1	[14, 18]	2.7
3	Nancy	0	[13, 18]	Lyon	0	[17, 20]	2.8
4	Dijon	0	[12, 14]	Tolouse	1	[3, 6]	1.5
5	Orleans	0	[13, 16]	Nantes	2	[13, 16]	4
6	Bordeaux	1	[2, 4]	Paris	1	[8, 14]	2.2
7	Tolouse	1	[17, 19]	Lyon	1	[21, 23]	2.7
8	Marseille	0	[4, 5]	Reims	2	[12, 15]	4.9
9	Lyon	0	[19, 21]	Dijon	1	[22, 23]	2.3
10	Paris	2	[12, 14]	Bordeaux	3	[20, 23]	2.8
11	Orleans	1	[2, 4]	Marseille	3	[13, 17]	1.6
12	Nantes	2	[18, 20]	Tolouse	2	[14, 17]	1.8
13	Tolouse	0	[13, 16]	Reims	1	[15, 21]	5.3
14	Paris	1	[13, 15]	Lyon	2	[6, 9]	3.1
15	Lyon	0	[4, 9]	Nancy	0	[3, 6]	2
16	Orleans	3	[19, 21]	Bordeaux	4	[22, 23]	0.8

Tabla A.31: Pedidos

A.8.2. Resultados

Configuración	Solución			Tiempos (seg.)		# Iteraciones		# Rutas	
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	Total
1	262,7	0,387134	262,7	1965	14400	174	26	0	200
2	262,7	0,399907	262,7	2844	14400	45	5	8114	8164
3	262,7	0,240513	262,7	3018	14400	170	30	0	200
4	262,7	0,405405	262,7	26214	14401	199	1	13955	14155
5	262,7	0	262,7	28801	1	6	1	0	7
6	262,7	0	262,7	28802	1	1	0	758	759
7	262,7	0	262,7	28801	1	4	1	0	5
8	167,6	0	262,7	28801	1107	1	1	5857	5859
9	262,7	0,452227	262,7	27285	14400	192	8	0	200
10	262,7	0,121067	262,7	28801	14400	8	0	11102	11110
11	262,7	0	262,7	28802	1	2	0	0	2
12	262,7	0	262,7	28802	1	1	0	47	48

Tabla A.32: Instancia 8 - resultados

A.9. Instancia 9

A.9.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Mexico DF	2	4
Guadalajara	0	1
San Luis Potosi	1	2
Chilpancingo	1	2
Tampico	0	1
Toluca	1	2
Veracruz	1	1
Oaxaca	1	1
Morelia	1	1
Colima	1	1

Tabla A.33: Cantidad de camiones y camioneros por ciudad en el momento inicial

- Horizonte de planificación: $T = 6$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 2{,}2$
- Costo de viaje por unidad de tiempo: $\alpha = 1$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

	Mexico DF	Guadalajara	San Luis Potosi	Chilpancingo	Tampico	Toluca	Veracruz	Oaxaca	Morelia	Colima
Mexico DF	0.0	5.4	4.4	2.75	5.8	0.75	4.0	5.6	3.1	7.3
Guadalajara	5.4	0.0	4.0	8.0	8.75	4.9	9.3	10.9	2.9	2.3
San Luis Potosi	4.4	4.0	0.0	6.9	4.8	4.5	8.1	9.5	4.6	6.6
Chilpancingo	2.75	8.0	6.9	0.0	8.6	3.25	6.6	6.8	5.7	10.0
Tampico	5.8	8.75	4.8	8.6	0.0	6.5	5.9	10.1	8.0	11.3
Toluca	0.75	4.9	4.5	3.25	6.5	0.0	4.6	6.2	2.5	6.8
Veracruz	4.0	9.3	8.1	6.6	5.9	4.6	0.0	4.4	6.9	11.5
Oaxaca	5.6	10.9	9.5	6.8	10.1	6.2	4.4	0.0	8.4	12.0
Morelia	3.1	2.9	4.6	5.7	8.0	2.5	6.9	8.4	0.0	5.0
Colima	7.3	2.3	6.6	10.0	11.3	6.8	11.5	12.0	5.0	0.0

Tabla A.34: Tiempo de viaje entre ciudades (en horas)

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Mexico DF	0	[7, 10]	San Luis Potosi	0	[20, 23]	4.5
2	Guadalajara	0	[6, 10]	Chilpancingo	1	[15, 21]	3.2
3	San Luis Potosi	0	[13, 18]	Morelia	0	[17, 20]	2
4	Colima	0	[10, 14]	Toluca	1	[4, 10]	2.5
5	Tampico	0	[11, 16]	Veracruz	2	[11, 16]	4
6	Toluca	1	[2, 7]	Oaxaca	1	[8, 16]	2.2
7	Oaxaca	1	[17, 22]	Mexico DF	1	[19, 23]	2.7
8	Veracruz	0	[7, 17]	Guadalajara	0	[12, 19]	4.8
9	Toluca	0	[17, 21]	San Luis Potosi	1	[19, 23]	2.8
10	Tampico	2	[8, 14]	Chilpancingo	2	[15, 23]	4.5
11	Mexico DF	1	[6, 12]	Colima	2	[13, 20]	2.3
12	Mexico DF	2	[17, 23]	Toluca	2	[13, 17]	1.8
13	San Luis Potosi	0	[11, 16]	Veracruz	1	[13, 21]	5.3
14	Chilpancingo	1	[10, 17]	Oaxaca	2	[6, 18]	3.1
15	Guadalajara	0	[4, 9]	San Luis Potosi	0	[3, 6]	2
16	Colima	1	[6, 11]	Morelia	1	[14, 20]	2.8

Tabla A.35: Pedidos

A.9.2. Resultados

Configuración	Solución			Tiempos (seg.)		# Iteraciones		# Rutas	
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	Total
1	253,7	0,301537	253,7	5475	14401	187	13	0	200
2	253,7	0,329874	253,7	28801	14401	29	0	39757	39786
3	253,7	0,269216	253,7	11510	14400	178	22	0	200
4	253,7	0,32637	253,7	28801	14401	26	1	33278	33305
5	253,7	0	253,7	28801	4	12	0	0	12
6	**232,5**	0	253,7	28802	4	2	0	4634	4636
7	253,7	0,216397	253,7	28801	14400	104	2	0	106
8	253,7	0,182893	253,7	28802	14400	3	0	4274	4277
9	253,7	0	253,7	28801	187	13	0	0	13
10	253,7	0,221127	253,7	28801	14400	3	0	7498	7501
11	253,7	0,290037	253,7	28801	14400	186	1	0	187
12	253,7	0,156484	253,7	28801	14400	4	0	7104	7108

Tabla A.36: Instancia 9 - resultados

A.10. Instancia 10

A.10.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Warsaw	1	3
Lodz	1	1
Poznan	1	2
Wroclaw	1	2
Krakow	1	2
Rzeszow	1	2
Bialystok	1	2
Olsztyn	0	1
Gdansk	1	2
Szczecin	0	1

Tabla A.37: Cantidad de camiones y camioneros por ciudad en el momento inicial

- Horizonte de planificación: $T = 5$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 4,5$
- Costo de viaje por unidad de tiempo: $\alpha = 2$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

	Warsaw	Lodz	Poznan	Wroclaw	Krakow	Rzeszow	Bialystok	Olsztyn	Gdansk	Szczecin
Warsaw	0.0	2.4	3.9	5.4	4.3	4.5	2.75	3.2	5.0	6.8
Lodz	2.4	0.0	2.25	3.4	3.5	5.0	4.9	4.0	4.75	5.25
Poznan	3.9	2.25	0.0	3.0	5.5	7.25	6.7	5.0	4.0	3.2
Wroclaw	5.4	3.4	3.0	0.0	2.5	4.9	8.4	7.25	6.4	4.8
Krakow	4.3	3.5	5.5	2.5	0.0	2.75	7.25	7.5	8.3	7.3
Rzeszow	4.5	5.0	7.25	4.9	2.75	0.0	6.4	7.6	9.4	9.5
Bialystok	2.75	4.9	6.7	8.4	7.25	6.4	0.0	3.6	6.2	9.6
Olsztyn	3.2	4.0	5.0	7.25	7.5	7.6	3.6	0.0	2.5	7.0
Gdansk	5.0	4.75	4.0	6.4	8.3	9.4	6.2	2.5	0.0	5.4
Szczecin	6.8	5.25	3.2	4.8	7.3	9.5	9.6	7.0	5.4	0.0

Tabla A.38: Tiempo de viaje entre ciudades (en horas)

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Warsaw	1	[7, 10]	Krakow	2	[15, 20]	7.3
2	Lodz	0	[8, 10]	Szczecin	1	[19, 23]	4.25
3	Poznan	1	[13, 18]	Olsztyn	1	[17, 20]	3
4	Wroclaw	0	[9, 14]	Rzeszow	0	[6, 11]	4.5
5	Rzeszow	0	[9, 16]	Gdansk	2	[9, 16]	3.25
6	Bialystok	0	[2, 7]	Poznan	0	[8, 16]	2.5
7	Olsztyn	0	[17, 22]	Warsaw	0	[19, 23]	3.7
8	Gdansk	0	[9, 15]	Krakow	1	[12, 20]	8.5
9	Szczecin	0	[17, 21]	Wroclaw	1	[19, 23]	3.5
10	Krakow	2	[8, 12]	Warsaw	2	[15, 19]	6
11	Gdansk	0	[6, 12]	Szczecin	0	[13, 20]	3.8
12	Olsztyn	2	[17, 23]	Poznan	2	[13, 17]	1.8
13	Bialystok	0	[12, 16]	Rzeszow	1	[9, 18]	4.3
14	Lodz	1	[10, 17]	Bialystok	2	[6, 18]	3.1
15	Warsaw	0	[4, 10]	Wroclaw	0	[18, 22]	9.5
16	Lodz	1	[6, 11]	Warsaw	1	[14, 20]	4.8

Tabla A.39: Pedidos

A.10.2. Resultados

Configuración	Solución			Tiempos (seg.)		# Iteraciones		# Rutas	
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	Total
1	579,6	0,484645	579,6	3979	14400	194	6	0	200
2	579,6	0,486006	579,6	28005	14401	39	0	37469	37508
3	579,6	0,458592	579,6	10935	14401	176	24	0	200
4	579,6	0,486025	579,6	28001	14401	146	0	12925	13071
5	579,6	0	579,6	28801	52	10	0	0	10
6	579,6	0,107833	579,6	28802	14400	1	0	732	733
7	579,6	0	579,6	28802	46	10	0	0	10
8	579,6	0,424776	579,6	28801	14400	10	0	13189	13199
9	579,6	0	579,6	28801	2	7	0	0	7
10	579,6	0,238095	579,6	28801	14400	2	0	5792	5794
11	579,6	0	579,6	28801	177	17	1	0	18
12	579,6	0,420031	579,6	28801	14400	9	0	11426	11435

Tabla A.40: Instancia 10 - resultados

A.11. Instancia 11

A.11.1. Definición de la instancia

Ciudad	Cant. camiones	Cant. camioneros
Berlin	2	2
Hamburg	0	1
Bremen	1	1
Cologne	0	0
Frankfurt	1	2
Stuttgart	1	2
Munich	2	3
Nuremberg	1	2
Leipzig	1	2
Dresden	0	1

Tabla A.41: Cantidad de camiones y camioneros por ciudad en el momento inicial

- Horizonte de planificación: $T = 5$
- Cantidad de momentos diarios: $K = 24$
- Penalidad por viaje sin mercadería, por unidad de tiempo: $\beta = 4{,}5$
- Costo de viaje por unidad de tiempo: $\alpha = 2$
- Máxima cantidad de momentos que un camión puede estar detenido: $M' = 12$

	Berlin	Hamburg	Bremen	Cologne	Frankfurt	Stuttgart	Munich	Nuremberg	Leipzig	Dresden
Berlin	0.0	2.9	3.9	5.7	5.25	6.25	5.8	4.5	2.0	1.9
Hamburg	2.9	0.0	1.0	3.9	4.6	6.3	7.25	5.75	3.75	4.75
Bremen	3.9	1.0	0.0	3.0	4.25	6.1	7.0	5.4	3.6	4.4
Cologne	5.7	3.9	3.0	0.0	1.9	3.5	5.4	3.75	4.75	5.5
Frankfurt	5.25	4.6	4.25	1.9	0.0	2.0	3.8	2.4	3.75	4.25
Stuttgart	6.25	6.3	6.1	3.5	2.0	0.0	2.2	2.1	4.6	4.9
Munich	5.8	7.25	7.0	5.4	3.8	2.2	0.0	1.6	4.2	4.3
Nuremberg	4.5	5.75	5.4	3.75	2.4	2.1	1.6	0.0	2.6	3.0
Leipzig	2.0	3.75	3.6	4.75	3.75	4.6	4.2	2.6	0.0	1.25
Dresden	1.9	4.75	4.4	5.5	4.25	4.9	4.3	3.0	1.25	0.0

Tabla A.42: Tiempo de viaje entre ciudades (en horas)

Pedido	*Pick-up*			*Drop-off*			Penalidad diaria
	Ciudad	Día	Ventana	Ciudad	Día	Ventana	
1	Frankfurt	1	[7, 10]	Stuttgart	1	[15, 20]	7.3
2	Munich	0	[8, 10]	Cologne	1	[19, 23]	4.25
3	Bremen	0	[13, 18]	Nuremberg	1	[17, 20]	3
4	Leipzig	0	[9, 14]	Hamburg	0	[6, 11]	4.5
5	Berlin	0	[9, 16]	Dresden	0	[9, 16]	2.25
6	Stuttgart	2	[2, 7]	Bremen	2	[8, 16]	3.5
7	Frankfurt	1	[17, 22]	Hamburg	2	[19, 23]	3.7
8	Cologne	1	[9, 15]	Berlin	2	[12, 20]	6.5
9	Bremen	0	[17, 21]	Dresden	0	[19, 23]	3.5
10	Hamburg	2	[8, 12]	Leipzig	2	[15, 19]	5.4
11	Berlin	0	[6, 12]	Nuremberg	1	[13, 20]	3.8
12	Dresden	2	[17, 23]	Munich	3	[13, 17]	1.8
13	Leipzig	0	[12, 16]	Stuttgart	1	[9, 18]	1.3
14	Nuremberg	0	[10, 17]	Frankfurt	0	[6, 18]	2.1
15	Munich	0	[4, 10]	Cologne	0	[18, 22]	4.5
16	Berlin	1	[6, 11]	Munich	1	[14, 20]	4.8

Tabla A.43: Pedidos

A.11.2. Resultados

Configuración	Solución			Tiempos (seg.)		# Iteraciones		# Rutas	
	Mejor sol.	Gap	Sol. inicial	Generación	Resolución	Heurística	Exacto	RPB	Total
1	541,8	0,647195	541,8	13464	14401	195	5	0	200
2	541,8	0,629845	541,8	28801	14401	15	0	45439	45454
3	541,8	0,618125	541,8	11887	14400	176	24	0	200
4	541,8	0,651486	541,8	20008	14404	46	1	43471	43518
5	541,8	0	541,8	28801	41	10	0	0	10
6	541,8	0,54199	541,8	28801	14400	2	0	6777	6779
7	541,8	0	541,8	28802	38	10	0	0	10
8	541,8	0,339248	541,8	28801	14401	6	0	9404	9410
9	541,8	0	541,8	28801	2	2	0	0	2
10	541,8	0,443245	541,8	28801	14400	2	0	5192	5194
11	541,8	0	541,8	28801	196	17	0	0	17
12	541,8	0,321748	541,8	28801	14400	6	0	8019	8025

Tabla A.44: Instancia 11 - resultados