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

Técnicas de optimización aplicadas a problemas de distribución bajo condiciones de tráfico variable

Tesis presentada para optar al título de Doctor de la Universidad de Buenos Aires en el área de Ciencias de la Computación

Gonzalo Lera Romero

Director de tesis: Juan José Miranda Bront

Consejera de estudios: Isabel Méndez-Díaz

Lugar de Trabajo: Departamento de Computación, FCEyN, Universidad de Buenos Aires.

Buenos Aires, 19 de Septiembre de 2025

TÉCNICAS DE OPTIMIZACIÓN APLICADAS A PROBLEMAS DE DISTRIBUCIÓN BAJO CONDICIONES DE TRÁFICO VARIABLE

Las entregas de última milla representan la etapa final del proceso de distribución, que suele tener lugar cerca de las ubicaciones de los clientes, dentro de ciudades grandes o pobladas. Tradicionalmente, estos problemas de distribución se han abordado en algunas variantes del conocido Problema de Ruteo de Vehículos (VRP), que consiste en visitar un conjunto de clientes con una flota de vehículos minimizando el costo operativo total. Desarrollar algoritmos efectivos para el VRP representa un desafío desde una perspectiva computacional, ya que pertenece a la clase de complejidad $\mathcal{NP}$-Hard. Frecuentemente, las condiciones de la red de transporte en estos tipos de escenarios varían a lo largo del día, ya que se acumula más congestión en ciertas horas. El Problema de Ruteo de Vehículos con Tiempos de Viaje Variables (TDVRP) es una generalización del VRP clásico, donde la velocidad de viaje no se asume constante a lo largo del horizonte de planificación y, como consecuencia, los tiempos de viaje varían en función del tiempo. En este sentido, se espera que el resultado de estos tipos de modelos sea más preciso, con soluciones más cercanas a las restricciones de la vida real, pero a expensas de algoritmos más complejos. En esta tesis, analizamos el impacto de incluir explícitamente la dependencia temporal en una amplia gama de variantes del TDVRP. La primera contribución aborda el Problema del Viajante de Comercio con Tiempos de Viaje Variables (TDTSP), una generalización del clásico Problema del Viajante de Comercio (TSP). Proponemos una formulación de Programación Lineal Entera Mixta (MILP), reforzada con nuevas familias de desigualdades válidas que capturan eficazmente los aspectos temporales del problema, y que es posible de ser adaptada a diferentes variantes.

La segunda contribución presenta un método de solución alternativo para el TDTSP basado en programación dinámica. Este enfoque logra mejoras significativas en el tiempo de ejecución y resuelve con éxito varias instancias que permanecían abiertas en la literatura.

La tercera contribución, en contraste, se centra en una variante multi-vehículo, el Problema de Ruteo de Vehículos con Tiempos de Viaje Variables y Vehículos Eléctricos (TDEVRP), que combina el ruteo dependiente del tiempo con restricciones operativas específicas de vehículos eléctricos, como el alcance de conducción limitado y el consumo de batería dependiente de la velocidad. Desarrollamos un algoritmo de branch-cut-and-price (BCP) que incorpora varios componentes de última generación. Experimentos computacionales extensos demuestran su efectividad y competitividad.

Palabras clave: Time Dependent, Problema de Ruteo de Vehículos, Problema del Viajante de Comercio, Vehículos Eléctricos, Problema del Camino más Corto Elemental, Branch Cut and Price, Labeling

OPTIMIZATION TECHNIQUES APPLIED TO DISTRIBUTION PROBLEMS UNDER VARIABLE TRAFFIC CONDITIONS

Last-mile deliveries account for the final stage of the distribution process, which typically takes place near customer locations, inside large or populated cities. Traditionally, these distribution problems have been tackled in some variants of the well-known Vehicle Routing Problem (VRP) consisting of visiting a set of customers with a fleet of vehicles while minimizing the total operational cost. Developing effective algorithms for the VRP represents a challenge from a computational perspective, as it belongs to the $\mathcal{NP}$-Hard complexity class. Frequently, the conditions of the transportation network in these types of scenarios vary along the day, as more congestion builds at certain hours. The Time-Dependent Vehicle Routing Problem (TDVRP) is a generalization of the classical VRP where the travel speed is not assumed to be constant throughout the planning horizon and, as a consequence, travel times vary throughout the planning horizon. In this sense, the outcome of these types of models is expected to be more accurate, with solutions closer to real-life constraints, but at the expense of more complex algorithms. In this thesis, we analyze the impact of explicitly including time dependency across a wide range of variants of the TDVRP. The first contribution addresses the Time-Dependent Traveling Salesman Problem (TDTSP), a generalization of the classical Traveling Salesman Problem (TSP). We propose a flexible Mixed Integer Linear Programming (MILP) formulation, reinforced with novel families of valid inequalities that effectively capture the temporal aspects of the problem.

The second contribution presents an alternative solution method for the TDTSP based on dynamic programming. This approach achieves significant improvements in execution time and successfully solves several instances that remained open in the literature.

The third contribution focuses on the Time-Dependent Electric Vehicle Routing Problem (TDEVRP), which combines time-dependent routing with operational constraints specific to electric vehicles, such as limited driving range and speed-dependent battery consumption. We develop a branch-cut-and-price (BCP) algorithm that incorporates several state-of-the-art components. Extensive computational experiments demonstrate its effectiveness and competitiveness.

Keywords: Time Dependent, Vehicle Routing Problem, Traveling Salesman Problem, Electric Vehicles, Elementary Shortest Path Problem, Branch Cut and Price, Labeling

AGRADECIMIENTOS

Me acuerdo que llegaba a casa a las once de la noche después de cursar, y encontraba a mi mamá esperándome con la cena lista, lista también para cenar conmigo y compartir un rato antes de terminar el día.

Me acuerdo que mi papá se levantaba cada mañana para llevarme en auto a la facultad durante los primeros años, cuando todavía todo era nuevo y el recorrido se hacía mejor acompañado.

Me acuerdo que mi hermano se sentaba conmigo durante horas para explicarme lo que él ya dominaba, incluso cuando tenía sus propias cosas que estudiar.

Me acuerdo que el amor de vida, Sofi, fue un apoyo fundamental haciendome compañia en esas noches en que el trabajo del doctorado se hacía eterno, y festejaba cada vez que le contaba que había avanzado, aunque fuera con un solo párrafo. Y también que Frida, mi gata, supervisaba todo desde el teclado con la seriedad de un jurado de tesis.

Me acuerdo de todas las tardes que pasamos con Juanjo frente al pizarrón, los almuerzos, los cafés y los intercambios de ideas; sus consejos, y la forma en que me enseñó a ser un mejor mentor (y persona) con el ejemplo.

Me acuerdo de la buena predisposición de todos los profesores y ayudantes, y de mis años de docencia en la mejor carrera que puede existir. Especialmente de Fran Soulignac, una de las personas más brillantes (y buenas) que conocí, con quien tuve la suerte de colaborar. De Fede, Brian y Monti, por haber compartido un rato de oficina y muchísimo tiempo de charlas. Y de Isa, por estar siempre atenta a todo lo que necesitaba para que este proyecto fuera posible.

Y por último, me acuerdo de todos esos momentos en los que mis amigos se alegraron por mis logros, y me motivaron a seguir adelante y alcanzar todo lo que me propuse.

Como suele decirse: ¡así cualquiera!

A mi tío Hugo, que celebraba cada uno de mis logros como si fueran suyos.

Índice general

1.. Introducción 1
1.1. El problema de ruteo de vehículos 1
1.2. Contribuciones 5
1.3. Resultados científicos 6

2.. Preliminares 9
2.1. Programación lineal 9
2.2. Generación de columnas 9
2.3. Programación lineal entera mixta 11
2.4. Algoritmos para problemas MILP 12
2.5. El modelo de dependiente del tiempo 16
2.6. Caso de estudio 1: El Problema del Viajante de Comercio 21
2.7. Caso de estudio 2: El Problema de Ruteo de Vehículos con Capacidades 22

3.. Un algoritmo branch and cut para el TDPTPRC 27
3.1. Introducción 27
3.2. Definición del problema 29
3.3. Formulaciones ILP 30
3.4. Desigualdades válidas 34
3.5. Algoritmo branch-and-cut 42
3.6. Resultados computacionales 44
3.7. Conclusiones 51
3.A. Apéndice 53

4.. Programación dinámica para el TDTSPTW 57
4.1. Introducción 57
4.2. Revisión de la literatura 58
4.3. Labels parciales y el criterio de dominancia parcial: una visión general 59
4.4. Algoritmo exacto de programación dinámica 61
4.5. Técnicas de preprocesamiento 67
4.6. Cálculo de cotas de finalización 68
4.7. Algoritmos Desarrollados 71
4.8. Resultados computacionales 72
4.9. Conclusiones e Investigación Futura 75
4.A. Apéndice 77

5.. Algoritmo de Branch-Cut-and-Price para el TDEVRP 83
5.1. Introducción 83
5.2. Problema de ruteo de vehículos eléctricos con tiempos de viaje variables 87
5.3. Un framework general para tiempos dependientes del tiempo 93
5.4. Algoritmo exacto 95
5.5. Configuración de branch-and-price 102
5.6. Experimentos computacionales 102
5.7. Conclusiones 109
5.A. Apéndice 112

6.. Conclusiones y Trabajo Futuro 115
6.1. Conclusiones Generales 115
6.2. Trabajo Futuro 116

1. INTRODUCCIÓN

1.1. El problema de ruteo de vehículos

La Investigación Operativa (IO) es una disciplina que tiene como objetivo principal el análisis y la generación de procedimientos para ayudar a las organizaciones en la toma de decisiones y otros problemas relacionados. Cuando estos problemas tienen un conjunto discreto de soluciones, la mejor se puede obtener mediante una inspección completa de ese conjunto. Sin embargo, para la mayoría de los escenarios, analizar cada solución no es práctico, debido al gran número de soluciones generalmente denominado explosión combinatoria. En esta dirección, las técnicas de Optimización Combinatoria son útiles para reducir el número de soluciones exploradas mientras se preserva cierta garantía de su calidad.

Uno de los sectores más desafiantes donde la aplicación de técnicas de Investigación Operativa ha producido mejoras significativas es el llamado Gestión de la Cadena de Suministro (SCM, por sus siglas en inglés). Este término se refiere al diseño, gestión y control de las operaciones necesarias para suministrar a los clientes un flujo de bienes (o servicios) de la manera más efectiva. La getión de la cadena de suministro es crucial para las empresas, ya que optimiza los procesos desde el procesamiento de materias primas hasta la entrega del producto. Mejora la eficiencia, reduce costos y mejora el rendimiento general. Como resultado, permite respuestas rápidas a las demandas del mercado, minimiza los costos de inventario y fomenta la colaboración.

La mayoría de las veces, las instalaciones de producción (por ejemplo, fábricas y granjas), donde tiene lugar la manufactura, se encuentran ubicadas fuera de las grandes ciudades, principalmente debido a los altos costos dentro de ellas. La entrega de la *primera milla* se encarga del transporte de mercancías desde su origen (por ejemplo, fábricas, granjas, o tiendas minoristas) hacia los centros de distribución cercanos a las ciudades. Estos centros tienen múltiples propósitos, como almacenamiento, empaquetado y sirven como ubicaciones intermedias, para el cross-docking, consolidación e intercambio de cargas entre vehículos. Al transportar mercancías desde las instalaciones de producción, los vehículos considerados para la entrega de la primera milla suelen ser camiones de gran capacidad o incluso trenes de carga. Estos vehículos ofrecen precios más bajos debido a su gran capacidad de carga. Sin embargo, tienen limitaciones para circular en ubicaciones específicas, como centros de ciudades, áreas residenciales y otras ubicaciones relacionadas dentro de una ciudad, debido a sus dimensiones y otras regulaciones, lo que resulta en la necesidad de llegar a los centros de distribución para descargar sus productos. Considerando esto, la entrega de *última milla* consiste en enviar mercancías desde los centros de distribución a las ubicaciones de los clientes, generalmente ubicados dentro de ciudades y áreas altamente pobladas. En la entrega de última milla, se deben tener consideraciones especiales. Las entregas están fragmentadas para que puedan ser realizadas por vehículos más pequeños, teniendo en cuenta el rango horario de entrega establecido de los clientes y la congestión del tráfico, lo que hace que el problema de ruteo de vehículos hacia los clientes sea más desafiante. Según diferentes estudios, uno de los factores clave radica en las entregas de última milla, alcanzando en algunos casos casi el 50 % del costo total de entrega de paquetes (ver, por ejemplo, Savelsbergh et al. [18] y Joerss et al. [9]).

Durante las últimas décadas, se ha explorado una amplia gama de problemas de distribución desde perspectivas tanto teóricas como prácticas (véase, por ejemplo, Toth et al. [19]). Esta extensa investigación ha dado lugar al desarrollo de numerosas técnicas de optimización, algoritmos y heurísticas diseñados para abordar desafíos logísticos complejos, resultando en herramientas sofisticadas con un impacto sustancial en las operaciones del mundo real.

El Problema de Ruteo de Vehículos (VRP) es una familia de problemas donde el objetivo es el diseño de planes de distribución para la entrega de mercancías a ubicaciones de clientes. Este problema es amplio y se aplica a diversas industrias, abarcando alimentos, textiles, contenedores, entre otros. El VRP fue introducido por primera vez por Dantzig et al. [6] como un enfoque para resolver un problema real de distribución. En su forma más simple, el VRP asume que se dispone

de una flota ilimitada de vehículos homogéneos, y representa la red de transporte con un digrafo completo $D = (V = \{0, 1, \cdots, n\}, A)$. El vértice 0 representa el depósito donde todas las rutas deben comenzar y terminar, y el resto de los vértices denotan las ubicaciones de los clientes. Además, cada arco $(i, j) \in A$ representa el camino directo entre las ubicaciones i y j y tiene una distancia asociada d_{ij} y un costo c_{ij}. Un conjunto de rutas es una solución factible solo cuando cada cliente es visitado exactamente una vez, y también cuando cada vehículo parte y termina en el depósito. Finalmente, el objetivo es minimizar el costo total de las rutas (la suma de los costos de los arcos).

Restricciones operativas Existe un gran número de aplicaciones reales para el VRP, cada una con restricciones y objetivos comerciales específicos. Debido a su relevancia práctica, el VRP generalmente se enriquece con restricciones operativas que acercan el modelo a las situaciones que se abordan. Por ejemplo, una de las variantes más estudiadas es el VRP con Capacidades (CVRP, por sus siglas en inglés) que considera que cada vehículo tiene una capacidad limitada de Q medida en el número de unidades a transportar, y cada cliente $i \in V$ tiene una demanda asociada q_i medida en la misma unidad. Por lo tanto, la demanda para una secuencia de clientes visitados por un solo vehículo no debe exceder su capacidad total. Otra variante ampliamente estudiada es el VRP con Ventanas de Tiempo (VRPTW), que incorpora una restricción operativa relacionada con la calidad del servicio de las rutas desde la perspectiva del cliente. Para cada cliente $i \in V$, una *ventana de tiempo* representa un intervalo de tiempo $[a_i, b_i]$ dentro del horizonte de planificación en el cual los clientes deben ser visitados para recoger o entregar los paquetes. Entonces, una ruta es factible si todos los clientes son visitados dentro de su correspondiente ventana de tiempo. Cada arco $(i, j) \in A$ tiene un tiempo de viaje asociado t_{ij} que indica el tiempo que tarda un vehículo en recorrer ese arco. Aunque en un contexto real el tiempo de viaje puede variar dependiendo de la hora de salida (a menudo referido como la naturaleza dependiente del tiempo del problema), en el VRPTW esto se simplifica asumiendo una velocidad constante del vehículo $v = 1$ durante todo el horizonte de planificación, lo que resulta en tiempos de viaje constantes (es decir, $t_{ij} = \frac{d_{ij}}{v}$).

Una de las variantes más estudiadas del VRP en la comunidad de IO es el conocido Problema del Viajante de Comercio (TSP), donde se considera un sólo vehículo con capacidad infinita. Este escenario surge de manera natural cuando todos los paquetes ya están asignados a los camiones de reparto, y por lo tanto el conjunto de clientes es conocido pero se necesita determinar el orden en que deben ser visitados para minimizar la distancia total recorrida (o el costo). El TSP es uno de los problemas más estudiados en Ciencias de la Computación (CS, por sus siglas en inglés) tanto por su relevancia práctica como por su característica teórica, lo que ha llevado a contribuciones científicas fundamentales en el campo de la investigación operativa y CS. El primer enfoque que consideró un método de planos de corte (Dantzig et al. [5]), la conocida heurística 2-opt (Lin et al. [15]) y el algoritmo CONCORDE desarrollado por (Applegate et al. [3]) representan contribuciones seminales para el campo, que han conducido a desarrollos sustanciales para muchos otros problemas y áreas relevantes. En esta línea, observar que el TSP puede ser adaptado de manera similar a la mayoría de las variantes consideradas para el VRP. Por ejemplo, el conocido TSP con Ventanas de Tiempo (TSPTW), consiste en encontrar una única ruta que visite a todos los clientes dentro de sus ventanas de tiempo.

Existe un gran número de aplicaciones en el mundo real donde visitar a cada cliente exactamente una vez puede resultar demasiado restrictivo. Por ejemplo, considere un servicio de entrega de productos perecederos, como productos frescos o suministros médicos. Supongamos que un camión de reparto tiene una capacidad limitada y necesita visitar un subconjunto de ubicaciones, cada una representando un cliente o un punto de entrega, dentro de un marco de tiempo especificado para garantizar la frescura de los productos. En este caso, la solución debe optimizar la ruta del camión de reparto, considerando factores como la naturaleza sensible al tiempo de los productos y la necesidad de maximizar el valor total de las entregas. El Orienteering Problem (OP) es un problema de optimización que busca encontrar la ruta más eficiente para un solo vehículo que visite un subconjunto de ubicaciones sin exceder las limitaciones de tiempo o distancia, mientras maximiza los ingresos totales o minimiza los costos.

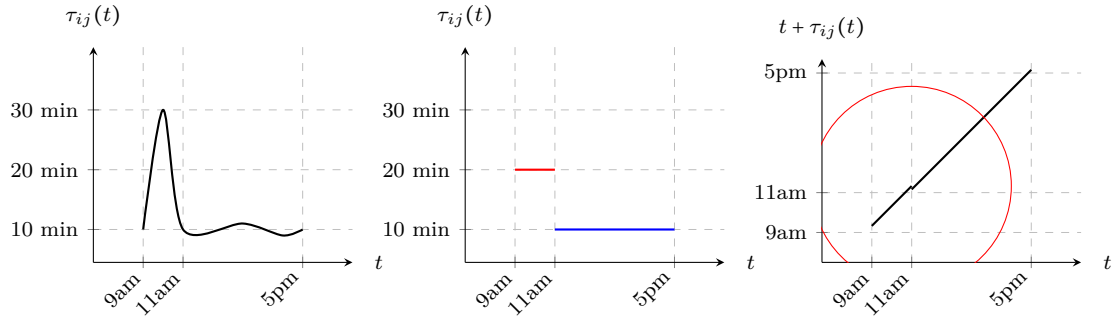
En algunos escenarios, los tipos de solicitudes son ligeramente diferentes a las entregas directas a

los clientes. El VRP con Recolecciones y Entregas (VRPPD, por sus siglas en inglés) es una variante del VRP clásico donde cada ruta implica tanto la recolección de artículos en ciertas ubicaciones como la entrega en otras. Como ejemplo, considere una plataforma de entrega de alimentos que ofrece servicios de comidas desde varios restaurantes hasta la puerta de los clientes. En este caso, las rutas deben optimizarse tanto para la recolección de comidas en los restaurantes (recolecciones) como para la entrega a los clientes (entregas). Cada recolección debe realizarse antes que su entrega asociada.

De manera similar, el VRP con Devoluciones (VRPB, por sus siglas en inglés) también considera recolecciones y entregas pero con restricciones ligeramente diferentes. Una devolución, en el contexto de la logística y el transporte, se refiere al transporte de mercancías en la parte de regreso del viaje de un vehículo. Implica la recolección o transporte de bienes o carga en la ruta de regreso al depósito de origen, típicamente después de completar la distribución principal de mercancías. Esto es particularmente relevante cuando se consideran cargas grandes y, por lo tanto, los vehículos necesitan ser vaciados antes de que comience la recolección para tener espacio de almacenamiento libre. Existe una amplia gama de aplicaciones del mundo real que pueden abordarse utilizando el VRPB, en particular en el contexto de la logística inversa. Por ejemplo, considere un centro de distribución minorista que suministra productos a una red de tiendas minoristas. En este escenario, el VRPB podría aplicarse para optimizar las rutas de los vehículos de entrega, teniendo en cuenta tanto las entregas de nuevos productos a un conjunto de tiendas como la posibilidad de recoger artículos devueltos o no vendidos para su retorno al centro de distribución desde un segundo conjunto de tiendas.

La flota de vehículos puede variar en diferentes contextos y una consideración cuidadosa de sus características puede mejorar en gran medida la calidad (y factibilidad) de las soluciones. En algunos escenarios, el número de camiones es limitado y expandir la flota no es una opción. Como resultado, algunos modelos incorporan explícitamente esta limitación. En otros contextos, la flota puede ser heterogénea, lo que significa que cada vehículo tiene diferentes características, como diferentes capacidades o tamaños que les permiten desplazarse por distintas secciones de la red de transporte. Por ejemplo, un camión grande no puede maniobrar fácilmente dentro de las calles estrechas de una red vial urbana. Una de las variantes que ha ganado relevancia en los últimos años es el llamado VRP con Vehículos Eléctricos (EVRP, por sus siglas en inglés). En la última década, muchas empresas han estado cambiando hacia alternativas de transporte más amigables con el medio ambiente. Como ejemplo, UPS planteó como objetivo que el 25 % de sus vehículos funcionen con combustible alternativo para 2020 (UPS [20]). Los gobiernos están introduciendo incentivos para estimular la adquisición y construcción de EVs, mientras que la Unión Europea también planea introducir sanciones por la emisión de CO_2 dentro del ruteo y la logística (EPEC 2011). En el EVRP, la flota de vehículos está compuesta por vehículos eléctricos (EVs) en lugar de los tradicionales vehículos con motor de combustión interna (ICEVs). Los EVs ofrecen varias ventajas sobre los ICEVs, como menor ruido y contaminación por CO_2, y generalmente menores costos en términos de energía. Sin embargo, una de las principales desventajas de los EVs es su autonomía restringida debido a la capacidad (todavía) limitada de sus baterías. En este sentido, el EVRP incorpora ubicaciones especiales llamadas estaciones de recarga donde los EVs pueden realizar recargas en ruta para evitar que la batería se agote. Este hecho aumenta la complejidad del problema desde una perspectiva algorítmica, pero ignorarlo puede resultar en soluciones inviables.

Dependencia temporal En las grandes ciudades, la congestión vial es altamente variable a lo largo del día. Existen horas pico usualmente asociadas con aquellos intervalos en los que las personas se desplazan hacia y desde sus lugares de trabajo. Como resultado, se vuelve esencial diseñar un mecanismo para incorporar esta información directamente en los modelos de VRP. Se han propuesto varios modelos para capturar el efecto de la congestión del tráfico en los tiempos de viaje. Uno de los primeros y más simples enfoques se denomina *costos dependientes de la posición*, que consiste en asignar un costo a cada arco dependiendo de su posición en una ruta dada (ver Abeledo et al. [1]). Más precisamente, un arco (i, j) visitado en la k-ésima posición de una ruta p, tiene un costo



(a) Ejemplo de patrones de tiempos de viaje en grandes ciudades

(b) Función de tiempo de viaje τ para la Figura 1.1a. El horizonte de 9 am a 5 pm se divide en dos intervalos (rojo y azul) para capturar las horas pico.

(c) Tiempos estimados de llegada $t + \tau(t)$. Los tiempos de salida donde no se cumple la propiedad FIFO están marcados en rojo.

Fig. 1.1: Ejemplo que ilustra patrones típicos de tiempos de viaje en grandes ciudades, y su representación considerando el modelo de Malandraki et al. [16].

c_{ij}^k. Esto permite un cierto nivel de granularidad para establecer los costos de los arcos asumiendo que la posición aproxima un intervalo del día en que se recorre el arco. La principal desventaja de este modelo radica en que la posición dentro de una ruta no siempre es una buena aproximación para la hora del día en que se visita el arco. De hecho, esto depende de la distancia de los arcos previos y de la velocidad del vehículo.

Otra aproximación es propuesta en Malandraki et al. [16], donde los tiempos de viaje (constantes) t_{ij} para el arco (i, j) se generalizan introduciendo una función escalonada $\tau_{ij}(t)$ que indica el tiempo de viaje para alcanzar j si se parte de i en el tiempo t. Estas funciones se definen dividiendo el horizonte de planificación en intervalos que representan los diferentes momentos del día, y luego se asigna un tiempo de viaje promedio para cada uno de ellos. Definir estos intervalos adecuadamente permite que el modelo capture las horas pico. Por ejemplo, considere un arco (i, j) donde los vértices i y j están ubicados en una zona residencial y una zona céntrica, respectivamente. Entonces, típicamente hay una hora pico en la mañana cuando las personas se desplazan al trabajo y luego el resto del día el tiempo de viaje disminuye. En este caso, el horizonte de planificación (de 9am a 5pm) puede dividirse en dos intervalos: de 9am a 11am, y de 11am a 5pm. Entonces, por ejemplo, $\tau_{ij}(t) = 20$ minutos cuando $t \leq 11$am, y $\tau_{ij}(t) = 10$ minutos en otro caso (ver Figura 1.1).

Uno de los problemas al considerar este enfoque para los tiempos de viaje es que la propiedad *first-in-first-out* (FIFO) no siempre se cumple. La propiedad FIFO establece que retrasar la salida de un vehículo nunca puede conducir a un tiempo de llegada más temprano (ver Figura 1.1c). En el ejemplo anterior, salir a las $t_1 = 10{,}59$am desde el vértice i conduce a un tiempo de llegada al vértice j de las 11.19am ($t_{ij}(10{,}59am) = 20$ minutos), sin embargo, esperando 2 minutos y saliendo a las $t_2 = 11{,}01$am el vehículo llega a las 11.11am ($t_{ij}(11{,}01am) = 10$ minutos).

Para mitigar esta situación, Ichoua et al. [8] proponen el modelo de *velocidades de viaje dependientes del tiempo*, que ha sido ampliamente aceptado en la literatura para VRPs con tiempos de viaje variables. Siguiendo una idea similar, el horizonte de planificación se divide en intervalos, pero en este caso, se considera una función escalonada de velocidad promedio para cada intervalo en lugar de tiempos de viaje promedio. Cada intervalo se denomina una *zona de velocidad*. Se calcula una función de tiempo de viaje $\tau_{ij}(t)$ basada en la distancia del arco y las velocidades correspondientes en cada intervalo que el vehículo atraviesa hasta llegar al destino. Notar que el vehículo puede viajar durante una o más zonas de velocidad hasta alcanzar su destino. Como resultado, estas funciones de tiempo de viaje son más complejas que las del modelo anterior, ya que son lineales a trozos en lugar de funciones escalonadas. Sin embargo, la ventaja es que satisfacen la propiedad FIFO y ofrecen aproximaciones más cercanas a escenarios de la vida real.

El VRP con Tiempos de Viaje Variables (TDVRP, por sus siglas en inglés) añade complejidad a las formulaciones tradicionales del VRP porque requiere modelar los tiempos de viaje variables de manera precisa y eficiente. Resolver el TDVRP típicamente implica una combinación de diseño innovador de algoritmos, estructuras de datos eficientes y técnicas de optimización adaptadas específicamente para problemas de ruteo con dependencia temporal. Investigadores y profesionales están continuamente trabajando para desarrollar nuevos enfoques que aborden estos desafíos y mejoren la efectividad de las soluciones para el TDVRP.

Objetivos Otro componente relevante de los VRPs radica en la función objetivo, que determina el valor de una solución factible y define métricas o indicadores relevantes a optimizar. Notar que una empresa puede considerar que el consumo de combustible es la métrica más crítica a minimizar, mientras que otra organización puede priorizar la satisfacción del cliente y, por lo tanto, minimizar los tiempos de entrega.

El VRP, en su forma original, considera que el recorrido de cada arco tiene un costo asociado y minimiza el costo de todos los arcos recorridos las rutas. Este costo puede ser cuidadosamente seleccionado para modelar muchas métricas a optimizar. Por ejemplo, si la variable a minimizar es el tiempo total de viaje, entonces es posible definir el costo c_{ij} de un arco $(i, j) \in A$ como el tiempo de viaje t_{ij}. En otras situaciones, se desea minimizar la distancia total recorrida debido a su impacto directo en la reducción del consumo de combustible; en este caso, el costo c_{ij} puede establecerse como la distancia d_{ij}. Otra función objetivo común consiste en minimizar el número total de vehículos requeridos. Esta métrica tiene un impacto directo en los costos fijos tanto en mantenimiento como en adquisición. Como último ejemplo, en términos prácticos, numerosas empresas se ven obligadas a subcontratar ciertas entregas a firmas externas, a menudo debido a restricciones como el tamaño limitado de la flota. En tales casos, el objetivo principal suele ser minimizar el número de entregas subcontratadas, ya que tienden a generar costos más altos.

En general, estos objetivos no son independientes y, a veces, se deben considerar más de uno al mismo tiempo, lo que conduce a un compromiso. De manera similar a las restricciones operativas, la inclusión de diferentes objetivos puede afectar el comportamiento y la aplicabilidad de técnicas algorítmicas específicas. De hecho, cada combinación de restricciones operativas y objetivos define una variación diferente del VRP y debe estudiarse por separado para obtener mejores resultados.

Finalmente, remitimos al lector a Toth et al. [19], para una revisión detallada de las variantes más estudiadas del VRP en la literatura, junto con una descripción de las técnicas algorítmicas más relevantes.

1.2. Contribuciones

Abordar la congestión dentro de los problemas de ruteo ha demostrado ser un desafío, principalmente debido a la creciente complejidad involucrada. Las formulaciones tradicionales de Programación Lineal Entera Mixta (MILP, por sus siglas en inglés) enfrentan problemas de escalabilidad a medida que aumentan el número de variables y restricciones con la introducción de la dimensión temporal. En consecuencia, a menudo tienen dificultades para resolver incluso instancias de tamaño pequeño de manera eficiente. Por otro lado, los enfoques de programación dinámica rápidamente enfrentan límites de recursos cuando no se cuentan con estructuras de datos y algoritmos cuidadosamente diseñados para resolver los pasos intermedios. Esta tesis examina a fondo estos desafíos y presenta soluciones de vanguardia para una variedad de problemas con tiempos de viaje dependientes del tiempo. A lo largo de esta tesis, consideramos el modelo propuesto por Ichoua et al. [8] cuando nos referimos a tiempos de viaje dependientes del tiempo.

Uno de los problemas con tiempos de viaje variables más estudiados es el TDTSP con restricciones de recursos (por ejemplo, ventanas de tiempo, capacidad). Antes de las contribuciones presentadas en este documento, la investigación para estos problemas se basaba principalmente en formulaciones MILP que integraban directamente la dependencia del tiempo mediante la introducción de nuevas variables para considerar el tiempo, junto con otras técnicas de optimización

como heurísticas de cota superior. Esto puede observarse en Adamo et al. [2] para el TDTSP, y en Arigliano et al. [4] y Montero et al. [17] para el TDTSPTW.

Esta tesis aborda varios problemas de ruteo mono-vehículo, incluyendo el TDTSP y el TDTSPTW. En primer lugar, se propone un enfoque novedoso que presenta una formulación compacta MILP con variables y restricciones reducidas, manteniendo a la vez los límites de relajación lineal establecidos en el trabajo previo de Montero et al. [17]. Además, se diseña un conjunto de nuevas familias de planos de corte para abordar la naturaleza dependiente del tiempo del problema. El algoritmo demuestra eficacia en la resolución de instancias que previamente no habían sido resueltas.

En investigaciones posteriores, se investiga un nuevo algoritmo de programación dinámica basado en labeling, con un enfoque particular en los detalles de implementación. En este trabajo, múltiples técnicas recientemente propuestas se adaptan al escenario con tiempos de viaje variables. La eficacia del nuevo algoritmo se evalúa en cinco conjuntos de datos distintos, que abarcan una amplia gama de restricciones y funciones objetivo extraídas de la literatura para el TDTSP, TDTSPTW y una variante del TSPTW donde el objetivo es minimizar la duración total de la ruta. Los resultados indican que este enfoque aborda eficazmente numerosos casos previamente no resueltos, al tiempo que mejora los tiempos promedio de ejecución.

Además del TDTSP, el TDVRPTW ha captado el interés de la comunidad de ruteo, y su relevancia ha sido particularmente pronunciada en los últimos años, impulsada por aplicaciones específicas como la gestión de flotas urbanas y el auge de tecnologías emergentes. Una de las extensiones naturales del TDVRPTW es el EVRPTW, que incorpora restricciones energéticas propias de los vehículos eléctricos. Los algoritmos exactos de última generación para este problema, sin embargo, operaban bajo la suposición de velocidad constante del vehículo (Desaulniers et al. [7]). El último capítulo de esta tesis propone un modelo de descarga de batería más complejo que considera explícitamente la variación de la velocidad del vehículo a lo largo del horizonte de planificación. Desde la perspectiva algorítmica, se presenta un algoritmo BCP con una nueva estrategia de ramificación que supera las reglas clásicas del EVRPTW. Desde un punto de vista gerencial, los resultados muestran que descuidar el impacto de la dependencia temporal podría conducir a costos más altos debido a rutas inviables.

Esta tesis está estructurada de la siguiente manera. El Capítulo 2 presenta una breve visión general de los principales conceptos asociados con MILP y algoritmos de labeling. Además, se discute una revisión general del modelo con tiempos de viaje variables y sus características más importantes. Finalmente, se presentan dos casos de estudio como ejemplo para ilustrar la aplicación de dichas técnicas. Estas ideas se utilizan a lo largo del resto de la tesis. Los siguientes tres capítulos presentan un conjunto de artículos académicos que abordan diversos problemas con tiempos de viaje variables utilizando diferentes técnicas. El Capítulo 3 presenta el nuevo enfoque del modelo MILP para el TDTSP con restricciones de recursos. En el Capítulo 4, se presenta y evalúa un algoritmo de programación dinámica para el TDTSP con restricciones de recursos sobre un conjunto de datos extenso. El Capítulo 5 se basa en estas contribuciones algorítmicas e introduce el novedoso problema TDEVRPTW y un marco algorítmico para incorporar múltiples variantes bien conocidas de la literatura. Finalmente, en el Capítulo 6, se presentan las conclusiones generales como resultado de las diversas investigaciones realizadas en los capítulos anteriores, y se discuten en detalle algunas líneas de trabajo futuro.

1.3. Resultados científicos

El contenido de esta tesis ha sido publicado como una serie de artículos académicos y presentado en conferencias tanto nacionales como internacionales. La siguiente lista incluye un resumen de la producción científica generada y una referencia a los capítulos específicos de este documento que están asociados con ellos.

- Publicaciones
    - *Integer programming formulations for the time-dependent elementary shortest path pro-*

blem with resource constraint Lera-Romero et al. [13] - Electronic Notes in Discrete Mathematics 2018 (Secciones 3.3.1 y 3.3.2).

- *A branch and cut algorithm for the time-dependent profitable tour problem with resource constraints* Lera-Romero et al. [10] - European Journal of Operational Research 2021 (Capítulo 3).
- *Dynamic programming for the time-dependent traveling salesman problem with time windows* Lera-Romero et al. [12] - INFORMS Journal on Computing 2022 (Capítulo 4).
- *A branch-cut-and-price algorithm for the time-dependent electric vehicle routing problem with time windows* Lera-Romero et al. [11] - European Journal of Operational Research 2024 (Capítulo 5).

- Conferencias internacionales
  - Euro/Alio 2018 - Bolonia, Italia
  - Verolog 2019 - Sevilla, España
  - International Conference on Computational Logistics 2020 - Twente, Países Bajos
  - 9th Informs Transportation Science and Logistics Society Workshop 2021 - Virtual, India
- Conferencias nacionales
  - Jornadas Argentinas de Informática (JAIIO) 2017 - Córdoba, Argentina
  - Jornadas Argentinas de Informática (JAIIO) 2018 - Córdoba, Argentina
  - Workshop in Operations, Networks and Data Analytics 2018 - Buenos Aires, Argentina
  - Workshop in Operations, Networks and Data Analytics 2019 - Buenos Aires, Argentina
- Otras publicaciones
  - *Linear edge costs and labeling algorithms: The case of the time-dependent vehicle routing problem with time windows* Lera-Romero et al. [14] - Networks 2020 (Realizado durante el doctorado pero no incluido en esta tesis).

Referencias

[1] Hernán Abeledo et al. «The Time Dependent Traveling Salesman Problem: Polyhedra and Branch-Cut-and-Price Algorithm». En: *Experimental Algorithms*. Ed. por Paola Festa. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, págs. 202-213.

[2] Tommaso Adamo et al. «An enhanced lower bound for the Time-Dependent Traveling Salesman Problem». En: *Comput. Oper. Res.* 113 (2020), pág. 104795.

[3] David L. Applegate et al. *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ, USA: Princeton University Press, 2007.

[4] Anna Arigliano et al. «Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm». En: *Discrete App. Math.* 261 (2018), págs. 28-39.

[5] G. Dantzig et al. «Solution of a Large-Scale Traveling-Salesman Problem». En: *J. Oper. Res. Soc. Am.* 2.4 (1954), págs. 393-410.

[6] G. B. Dantzig et al. «The Truck Dispatching Problem». En: *Management Science* 6.1 (1959), págs. 80-91.

[7] Guy Desaulniers et al. «Exact algorithms for electric vehicle-routing problems with time windows». En: *Oper. Res.* 64.6 (2016), págs. 1388-1405.

[8] Soumia Ichoua et al. «Vehicle dispatching with time-dependent travel times». En: *Eur. J. Oper. Res.* 144.2 (2003), págs. 379-396.

[9] Martin Joerss et al. *Parcel delivery: The future of last mile*. Inf. téc. McKinsey&Company, 2016.

[10] Gonzalo Lera-Romero et al. «A branch and cut algorithm for the time-dependent profitable tour problem with resource constraints». En: *European Journal of Operational Research* 289.3 (2021), págs. 879-896. ISSN: 0377-2217.

[11] Gonzalo Lera-Romero et al. «A branch-cut-and-price algorithm for the time-dependent electric vehicle routing problem with time windows». En: *European Journal of Operational Research* 312.3 (2024), págs. 978-995. ISSN: 0377-2217.

[12] Gonzalo Lera-Romero et al. «Dynamic Programming for the Time-Dependent Traveling Salesman Problem with Time Windows». En: *INFORMS Journal on Computing* 34.6 (2022), págs. 3292-3308.

[13] Gonzalo Lera-Romero et al. «Integer programming formulations for the time-dependent elementary shortest path problem with resource constraints». En: *Electronic Notes in Discrete Mathematics* 69 (2018). Joint EURO/ALIO International Conference 2018 on Applied Combinatorial Optimization (EURO/ALIO 2018), págs. 53-60. ISSN: 1571-0653.

[14] Gonzalo Lera-Romero et al. «Linear edge costs and labeling algorithms: The case of the time-dependent vehicle routing problem with time windows». En: *Networks* 76.1 (2020), págs. 24-53.

[15] S. Lin et al. «An Effective Heuristic Algorithm for the Traveling-Salesman Problem». En: *Operations Research* 21.2 (1973), págs. 498-516.

[16] Chryssi Malandraki et al. «Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms». En: *Transportation Science* 26 (1992), págs. 185-200.

[17] Agustín Montero et al. «An integer programming approach for the time-dependent traveling salesman problem with time windows». En: *Comput. Oper. Res.* 88 (2017), págs. 280-289.

[18] Martin W. P. Savelsbergh et al. «50th Anniversary Invited Article—City Logistics: Challenges and Opportunities». En: *Transport. Sci.* 50.2 (2016), págs. 579-590.

[19] Paolo Toth et al., eds. *Vehicle Routing: Problem, Methods and Applications*. Philadelphia, PA, USA: Society for Industrial y Applied Mathematics, 2014. ISBN: 0-89871-498-2.

[20] UPS. *Accelerating Sustainable Solutions, UPS 2019 Sustainability Progress Report*. 2019.

2. PRELIMINARES

2.1. Programación lineal

El problema de Programación Lineal (LP, por sus siglas en inglés) consiste en encontrar (si existe) una asignación a un conjunto de variables tal que se satisfaga un conjunto de restricciones mientras se maximiza (o minimiza) una función objetivo lineal. Formalmente, en un problema de maximización, dada una matriz $A \in \mathbb{R}^{m \times n}$ y vectores $b \in \mathbb{R}^m$ y $c \in \mathbb{R}^n$, el objetivo es encontrar una solución $x \in \mathbb{R}^n$ que satisfaga $Ax \leq b$ y maximice la función objetivo definida por $z(x) = cx$,

$$\max_{x \in \mathbb{R}^n} \{cx \mid Ax \leq b\}. \tag{2.1}$$

De manera similar, el problema de minimización asociado puede definirse como

$$\min_{x \in \mathbb{R}^n} \{c \mid Ax \geq b\}. \tag{2.2}$$

En general, la teoría detrás de LP radica en la observación de que el conjunto de soluciones factibles definido por $H = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ representa un poliedro (si está acotado). A continuación, presentamos un ejemplo de un modelo de LP y su representación del espacio de soluciones (o poliedro), en un sistema de coordenadas bidimensional.

Considere el siguiente problema de LP:

$$\begin{array}{ll} \text{mín} & z(x,y) = x + y \\ \textbf{sujeto a} & \\ & \frac{1}{3}x - y \leq -2 \\ & 2x - y \leq 5{,}5 \\ & \frac{1}{2}x + y \geq 2 \\ & 2x - y \geq 0{,}5 \\ & x, y \geq 0 \end{array} \tag{2.3}$$

Cada restricción es lineal y, por lo tanto, en el ejemplo define una línea en el plano y el área por encima (o por debajo) de ella. El poliedro asociado con LP_1 es el conjunto de puntos que satisfacen todas las restricciones al mismo tiempo. La Figura 2.1 ilustra las cuatro restricciones y el área que define el poliedro coloreada en verde. Además, la solución factible que optimiza la función objetivo es $(4{,}5, 3{,}5)$, con valor objetivo 8, denotado como z^*.

Vale la pena observar que el óptimo se encuentra sobre un extremo del poliedro, lo cual es un resultado fundamental de la programación lineal. En todo problema de LP acotado con al menos una solución factible, existe una solución óptima ubicada en un punto extremo del poliedro asociado.

Siguiendo esta idea, algunos algoritmos pueden resolver LP en tiempo polinomial (por ejemplo, Karmarkar [9]). En contraste, existen otros algoritmos que tienen una complejidad temporal exponencial en el peor caso, pero que funcionan de manera eficiente en la práctica. Uno de los algoritmos más famosos con estas características es el conocido algoritmo *Simplex* propuesto por Chvátal [1]. Hoy en día, existe una amplia gama de productos comerciales que implementan estos algoritmos de manera eficiente (por ejemplo, CPLEX, AMPL, Gurobi, por nombrar solo algunos), lo que permite que esta técnica se utilice frecuentemente para resolver problemas prácticos.

2.2. Generación de columnas

Cuando los modelos de LP tienen un número exponencial de columnas, considerarlas explícitamente es prohibitivo, incluso para algoritmos de solución en tiempo polinomial. Para abordar este problema, los algoritmos de *generación de columnas* (CG, por sus siglas en inglés) encuentran la

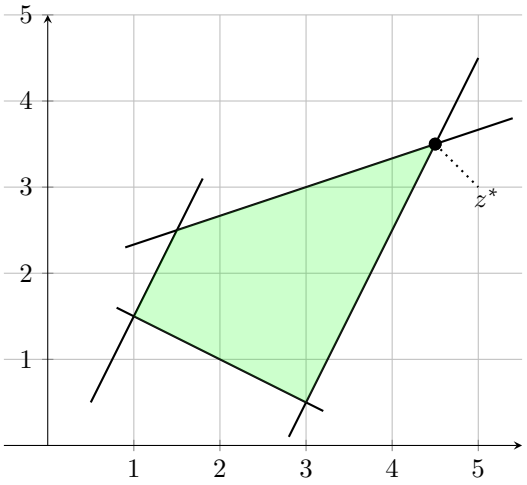


Fig. 2.1: Ejemplo del poliedro definido por LP_1.

solución óptima de un LP considerando solo un puñado de columnas bajo demanda. Para introducir esta técnica, primero necesitamos presentar el concepto de *dualidad*, que es fundamental para entender la motivación detrás de la CG.

Dado un problema de minimización $P := \text{mín}\{cx : Ax \geq b, x \geq 0\}$ con $A \in \mathbb{R}^{m \times n}$ y $x \in \mathbb{R}^n$, su problema *dual* se define como $D := \text{máx}\{by : A^t y \leq c, y \geq 0\}$ con $y \in \mathbb{R}^m$. Observar que cualquier problema de LP puede reescribirse en una *forma estándar*, es decir, como un problema de minimización con solo desigualdades mayores o iguales. El concepto de dualidad ha sido estudiado extensamente (p. ej. Chvátal [1]), dando origen a los teoremas de dualidad débil y fuerte.

Teorema 2.1 (Dualidad débil)**.** *Si $\bar{x}$ es una solución factible de P y $\bar{y}$ es una solución factible de D, entonces $c\bar{x} \geq b\bar{y}$.*

Teorema 2.2 (Dualidad fuerte)**.** *Si P tiene una solución óptima $\bar{x}$, entonces D tiene una solución óptima $\bar{y}$ y $c\bar{x} = b\bar{y}$.*

Cabe señalar que el teorema de dualidad fuerte implica que resolver cualquiera de las dos formulaciones es equivalente. Sin embargo, el número de variables y restricciones en las dos formulaciones difiere, ya que cada restricción en el dual corresponde a una variable en el primal, y cada restricción en el primal corresponde a una variable en el dual. Además, para cada variable en P (o columna de A), existe una restricción en D (o fila en A^t). Esta observación sirve como base para derivar el algoritmo CG.

Si la formulación P tiene un gran número de variables (columnas), entonces podemos considerar una versión restringida P' con solo un subconjunto de las variables. Formalmente, $P' := \text{mín}\{cx : A'x' \geq b, x' \geq 0\}$, donde A' es el subconjunto de columnas de A asociadas a aquellas variables consideradas $x' \subseteq x$. Adicionalmente, sea D' el problema dual asociado a P'. La ventaja de definir P' de esta manera radica en el hecho de que el número de columnas permanece manejable, y por lo tanto puede resolverse con un algoritmo genérico para LP y obtener su solución óptima x^*, así como una solución óptima y^* de D' de manera eficiente.

La solución x^* es óptima en P', y por lo tanto factible en P si se extiende con 0 en las variables que se dejaron fuera. Sin embargo, x^* no es necesariamente óptima en P. Por otro lado, y^* es una solución óptima en D', pero dado que D' tiene menos restricciones que el problema original D, potencialmente y^* no es factible en D. Si y^* es factible en D, es decir, no se viola ninguna restricción, entonces y^* es una solución óptima de D, y debido al Teorema 2.2, x^* debe ser óptima para P también. De lo contrario, existe una restricción definida por una fila de A^t (o columna de A) en D que es violada por y^* y que no fue considerada en D' (o P'). El problema de encontrar tal restricción se llama *problema esclavo* (SP, por sus siglas en inglés) o *problema de pricing*. Una

vez que esta restricción en el problema dual es identificada, se añade como una columna en A, los problemas P' y D' son actualizados, y se repiten los pasos anteriores. Generalmente, el problema original P se llama el *master problem* (MP, por sus siglas en inglés) y P' se llama el *restricted master problem* (RMP, por sus siglas en inglés). El Algoritmo 1 describe una visión general de los algoritmos CG.

Algorithm 1 Visión general de los algoritmos CG.

Entrada: Un problema LP P.

1. *Inicialización:* Definir un RMP P' con un subconjunto de las columnas de P.
2. *Reoptimización:* Resolver P' con un solver genérico de LP. Sea x^*, y^* las soluciones óptimas de P' y su problema dual. .
3. *Pricing:* Determinar si alguna restricción en D es violada por y^*. Si no hay ninguna, retornar x^*. De lo contrario, agregar a P' la variable de P que corresponde a la restricción violada en D, y repetir el Paso 2.

En general, debido a la estructura del problema estudiado, el problema de pricing tiene una interpretación individual y a menudo corresponde a un problema bien conocido de la literatura. Por ejemplo, en los modelos clásicos de generación de columnas para el Problema de Coloreo de Vértices (VCP, por sus siglas en inglés), el problema de pricing es el Problema de Conjunto Independiente con Máximo Peso. En el caso del VRP, el problema de pricing se convierte en el Problema de Camino Mínimo Elemental. Remitimos al lector a Desaulniers et al. [3] para ejemplos más detallados.

2.3. Programación lineal entera mixta

Aunque la programación lineal tiene una gran variedad de aplicaciones, existen muchos problemas que no pueden ser representados como un problema de LP. Esto se debe a que las variables en un LP pueden tomar cualquier valor real, lo que la hace inapropiada para modelar situaciones que involucran decisiones binarias (como elecciones sí/no) o que requieren que las variables representen cantidades enteras exactas que no pueden ser aproximadas por números reales. Cuando se consideran restricciones adicionales que imponen que los dominios de las variables sean enteros, el problema se denomina *Programación Lineal Entera* (ILP, por sus siglas en inglés), o Programación Lineal Entera Mixta (MILP, por sus siglas en inglés) si están presentes tanto variables continuas como enteras. Generalmente, los problemas más desafiantes del área de optimización combinatoria en la literatura pertenecen a esta categoría. Una formulación MILP tiene la siguiente estructura:

$$\begin{aligned} \text{mín} \quad & z(x) = cx \\ \textbf{sujeto a} \quad & \\ & Ax \geq b, \\ & x \geq 0, \\ & x_i \in \mathbb{R} \quad \forall i \in C, \\ & x_i \in \mathbb{Z} \quad \forall i \in I. \end{aligned} \tag{2.4}$$

Las variables se dividen en dos conjuntos: I, los índices de las variables que deben tomar valores enteros, y C, aquellas que pueden tomar valores continuos. La relajación lineal de este MILP se obtiene eliminando las restricciones de integralidad, resultando en el conjunto $P = \{x \in \mathbb{R}^n \mid Ax \geq b, x \geq 0\}$. Sea $z(x)$ la función objetivo y $\bar{z} = \text{mín}\{z(x) \mid x \in P\}$ como el valor óptimo de la relajación lineal. Dado que la región factible del MILP está contenida en P, se deduce que el valor óptimo $z(x^*)$ del MILP satisface $z(x^*) \geq \bar{z}$ en un problema de minimización. Por lo tanto, la relajación lineal proporciona una cota inferior válida.

Desde una perspectiva geométrica, sea $conv(S)$ la cápsula convexa de S, es decir, el poliedro más pequeño que incluye todos los puntos en S. Un resultado interesante de los modelos MILP indica

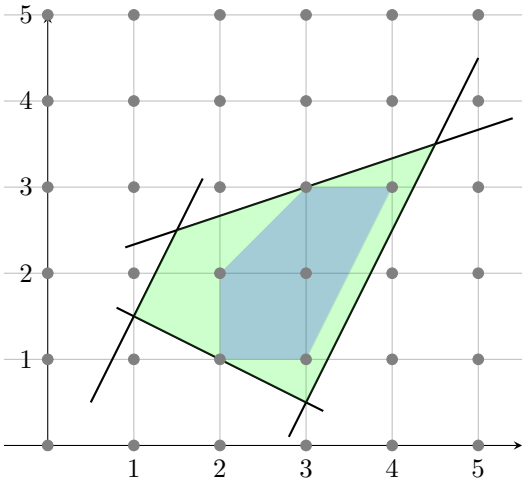


Fig. 2.2: Ejemplo: poliedro de relajación lineal P (verde), y cápsula convexa $conv(S)$ (azul).

que si $conv(S)$ puede ser representado con un conjunto de desigualdades de tamaño polinomial, entonces se puede encontrar una solución óptima usando un solver LP estándar. La razón es el hecho de que los puntos extremos de $conv(S)$ pertenecen a S, y un modelo LP tiene al menos un óptimo en sus puntos extremos, como se mencionó anteriormente.

A continuación, presentamos un ejemplo. Sea $MILP_1$ el siguiente problema MILP.

$$
\begin{array}{ll}
\text{mín} & z(x,y) = x + y \\
\textbf{sujeto a} & \\
& \frac{1}{3}x - y \leq -2 \\
& 2x - y \leq 5{,}5 \\
& \frac{1}{2}x + y \geq 2 \\
& 2x - y \geq 0{,}5 \\
& x, y \in \mathbb{N}
\end{array}
$$

La Figura 2.2 ilustra el poliedro de relajación lineal P asociado a $MILP_1$ (verde), y la cápsula convexa $conv(S)$ del conjunto de soluciones (azul).

En su forma general, el problema MILP pertenece a la clase de complejidad $\mathcal{NP}$-Hard, lo que significa que no se conocen algoritmos polinomiales que puedan resolver cualquier instancia de este problema (Garey et al. [5]). Sin embargo, existe una amplia variedad de métodos que abordan la resolución de formulaciones MILP de manera inteligente y logran resultados eficientes en la práctica. A continuación, presentamos una breve revisión de las técnicas más comunes consideradas en la literatura.

2.4. Algoritmos para problemas MILP

En esta sección, ofrecemos una visión conceptual de las principales familias de algoritmos exactos para resolver problemas MILP. La mayoría de estos algoritmos pertenecen a una de las siguientes categorías:

- Algoritmos de planos de corte,
- Algoritmos Branch-and-Bound (BB),
- Algoritmos Branch-and-Cut (BC),
- Algoritmos Branch-and-Price (BP).

2.4.1. Algoritmos de planos de corte

La idea detrás de esta técnica es añadir de forma incremental desigualdades válidas a la formulación MILP para ajustar el poliedro asociado a la relajación lineal sin excluir soluciones factibles, acercándolo a la cápsula convexa. Sea P el poliedro definido por la relajación lineal de una formulación MILP M, y sea S el conjunto de soluciones enteras factibles. Dado $\pi \in \mathbb{R}^m$ y $\pi_0 \in \mathbb{R}$, la desigualdad lineal $\pi x \geq \pi_0$ es válida para el modelo M si $\pi x \geq \pi_0$ para todo $x \in S$. Un plano de corte es una desigualdad válida que no es satisfecha por la solución óptima de la relajación lineal. Notar que siempre existe un plano de corte si el óptimo de la relajación lineal no es entero. El Algoritmo 2 define un marco general para procedimientos de planos de corte.

Algorithm 2 Visión general de los algoritmos de planos de corte.

Entrada: Un problema MILP M.

1. Resuelva la relajación lineal. Sea x^* el óptimo.
2. Si $x^* \in S$, detenerse.
3. *Separación:* Encuentre una desigualdad válida $\pi x \leq \pi_0$ violada por x^* e insértela en M.
4. Regrese al Paso 1.

Uno de los principales desafíos con esta técnica es la dificultad para encontrar planos de corte cercanos a la cápsula convexa de S. La razón detrás de esto es que encontrar planos de corte puede ser complicado tanto desde una perspectiva teórica como en términos de eficiencia computacional. En general, existen dos métodos principales para generar planos de corte.

El primer grupo se denomina *cortes de propósito general* que no dependen de un problema específico sino que, en cambio, se derivan del sistema de desigualdades en sí de manera general. Un ejemplo de esta clase de cortes son los bien conocidos Gomory Cuts Gomory [6], que toman información de la ejecución del algoritmo Simplex y la combinan con argumentos de integralidad. Para más detalles sobre esta familia de cortes, véase Nemhauser et al. [10]. Aunque estos cortes tienen la ventaja de ser generales y aplicables a la mayoría de los modelos, pueden no ser tan fuertes y por lo tanto tienen menos impacto en la práctica. En este sentido, otra línea de investigación consiste en idear familias válidas de desigualdades explotando información específica del problema. Ejemplos clásicos de este tipo de planos de corte son la familia de *Clique Cuts* para el *Problema de Conjunto Independiente* Rebennack [12] o las *Subtour Elimination Constraints* para el TSP Drexl et al. [4]. Generalmente, estas familias de desigualdades son críticas para mejorar el desempeño de los algoritmos en la práctica. Sin embargo, derivarlas requiere un entendimiento más profundo de los problemas en cuestión y no siempre son evidentes por sí mismas. Además, el esfuerzo dedicado a un problema específico puede no ser sencillo de extrapolar a problemas diferentes.

Para asegurar que los algoritmos de planos de corte terminen, es necesario garantizar que si el óptimo de la relajación lineal es fraccionario, exista al menos una desigualdad válida (corte) que lo separe. Además, la convergencia a una solución entera debería idealmente ocurrir en un número finito de pasos. Aunque existen procedimientos que proporcionan tales garantías en teoría, los algoritmos de planos de corte a menudo no se usan de forma aislada en la práctica, sino que se combinan con la técnica de *Branch-and-Bound* para mejorar la robustez y la eficiencia.

2.4.2. Algoritmos branch-and-bound

Otra técnica que ha demostrado ser efectiva para resolver problemas MILP es el llamado *Branch and Bound* (BB, por sus siglas en inglés). Este método explora sistemáticamente el espacio de soluciones, con el objetivo de evitar enumerar soluciones innecesarias mediante el uso de cotas para podar regiones subóptimas. El resto de esta sección considera un problema de minimización, pero todas las ideas pueden adaptarse fácilmente para manejar el caso de maximización.

El esquema general sigue la técnica de *Divide & Conquer* y consiste en construir un árbol de enumeración donde el nodo raíz representa el problema MILP original, y las ramas resultan de una partición del espacio de búsqueda. Esencialmente, cada nodo representa un subproblema que corresponde a encontrar el óptimo dentro de una región restringida. Para acelerar este proceso, se diseñan reglas de factibilidad y optimalidad para reducir el tamaño del *árbol de BB*. Esta última regla se denomina generalmente *Bounding*.

La regla de poda por factibilidad surge del análisis del poliedro asociado al nodo actual. Si este está vacío, entonces no existe solución factible en esta rama, y puede ser descartada. Por otro lado, la regla de poda por optimalidad consiste en calcular una cota inferior para el subproblema actual, y si esta es mayor o igual a alguna cota superior conocida para el problema original, entonces se puede detener la exploración del nodo. Típicamente, la cota superior está dada por la mejor solución conocida para el problema. La cota inferior puede obtenerse encontrando la solución óptima de la relajación lineal del subproblema actual. En algunos problemas, esta cota es demasiado débil y, por lo tanto, aún se enumeran un gran número de nodos. Algunos métodos alternativos para calcular cotas inferiores incluyen la llamada *Relajación Lagrangeana*, o algoritmos diseñados para problemas específicos.

El *branching* de un nodo consiste en particionar el espacio de búsqueda de soluciones en dos o más subproblemas, de tal manera que se preserven todas las soluciones enteras. Cada uno de estos subproblemas está representado por un nodo hijo que resulta de insertar una o más restricciones al nodo padre. Una de las reglas de branching más comunes se denomina *variable branching*. Después de resolver la relajación lineal, la idea es encontrar una variable x_i con valor fraccionario f en la solución óptima, y dividir el problema original en dos subproblemas nuevos y más pequeños añadiendo las restricciones $x_i \leq \lfloor f \rfloor$ y $x_i \geq \lceil f \rceil$, respectivamente. Si se puede seleccionar más de una variable fraccionaria, se establece un criterio de *selección de variable*. Los enfoques estándar incluyen el *minimizar la infactibilidad*, *maximizar la infactibilidad*, *pseudo-costos* y *strong-branching*, entre otros (véase Nemhauser et al. [10]). Estos suelen implementarse e incluirse en herramientas comerciales.

La visión general de un algoritmo BB se describe en el Algoritmo 3. Vale la pena observar que la técnica BB presenta un marco general, pero una gran parte de las decisiones dependen del problema a resolver. En este sentido, la elección correcta de los parámetros del algoritmo es crucial para la eficiencia y, por lo tanto, la experimentación y el análisis se convierten en un paso fundamental al construir este tipo de algoritmo.

Algorithm 3 Esquema general para algoritmos BB.

Entrada: Un problema MILP.

1. *Inicialización:* Crear una cola Q con el nodo raíz. Inicializar $z_{UB} = \infty$.
2. *Selección de nodo:* Si $Q = \varnothing$ retornar z_{UB}. De lo contrario, extraer un nodo N de Q.
3. *Acotación:* Resolver la relajación lineal de N.
   Si es infactible, regresar al Paso 2. De lo contrario, sea $\bar{x}$ el óptimo de la relajación y $\bar{z}$ el valor objetivo.
   - Si $\bar{x}$ es factible para el problema original, entonces establecer $z_{UB} = \min(z_{UB}, \bar{z})$. Regresar al Paso 2.
   - Si $\bar{x} > z_{UB}$, entonces el nodo puede ser acotado (optimalidad). Regresar al Paso 2.
4. *Branching:* Generar subproblemas $N_1, \cdots, N_k$ para el nodo actual N y agregarlos a Q.

2.4.3. Algoritmos de branch-and-cut

Tanto los algoritmos de BB como las técnicas de planos de corte son efectivos bajo diferentes

condiciones. Los algoritmos BB dependen en gran medida de la calidad de la relajación lineal; si es débil, el acotamiento es menos efectivo, lo que lleva a una enumeración excesiva. Por otro lado, los métodos de planos de corte dependen de la disponibilidad de familias de desigualdades válidas que pueden separar soluciones fraccionarias. Su rendimiento también depende de la eficiencia del procedimiento de separación y del número de iteraciones requeridas.

Naturalmente, combinar estas dos técnicas en un nuevo método llamado *Branch and Cut* (BC) permite mitigar los problemas mencionados anteriormente. Un algoritmo pionero de este tipo es propuesto en Crowder et al. [2], donde se considera un algoritmo de plano de corte con un criterio de parada determinado para mejorar las cotas proporcionadas por la relajación lineal. Después de terminar este paso, si no se encuentra una solución entera, entonces se ejecuta un algoritmo de BB sobre el modelo fortalecido con cortes. De esta manera, lograron una reducción dramática en el tamaño del árbol de BB. Hoy en día, la idea de incorporar cortes sólo en el nodo raíz se conoce como *Cut and Branch* (CB). Más tarde, en Grötschel et al. [7] y Padberg et al. [11] esta idea se generaliza aplicando un algoritmo de plano de corte en cada nodo del árbol de BB para obtener una reducción adicional en su tamaño.

Esta técnica es particularmente relevante en la fase de experimentación, ya que su eficiencia depende de un compromiso entre el peso dado al algoritmo del plano de corte contra la enumeración del algoritmo BB. Si se agregan demasiados planos de corte en cada nodo, la relajación lineal puede volverse cada vez más grande y computacionalmente costosa, provocando que las mejoras en la cota inferior disminuyan en relación con el tiempo dedicado al procedimiento de separación. Por otro lado, si no se dedica suficiente esfuerzo a mejorar las cotas inferiores, el árbol de enumeración puede volverse demasiado grande para manejar y, por lo tanto, llevar a tiempos de ejecución más altos. Por lo tanto, estos parámetros deben ajustarse cuidadosamente considerando cada problema en particular.

El Algoritmo 4 describe un marco general de BC. Análogamente a los algoritmos BB, hay una serie de decisiones (o parámetros) abiertos que deben definirse después de una cuidadosa experimentación.

Algorithm 4 Esquema general para algoritmos BC.

Entrada: Un problema MILP M.

1. *Inicialización:* Crear una cola Q con el nodo raíz. Inicializar $z_{UB} = \infty$.
2. *Selección de nodo:* Si $Q = \varnothing$ devolver z_{UB}. De lo contrario, sacar un nodo N de Q.
3. *Acotación:* Resolver la relajación lineal de n.
   Si es infactible volver al Paso 2. De lo contrario, sea $\bar{x}$ el óptimo de la relajación y $\bar{z}$ el valor objetivo.
   - Si $\bar{x}$ es factible para el problema original, entonces establecer $z_{UB} = \min(z_{UB}, \bar{z})$. Volver al Paso 2.
   - Si $\bar{x} > z_{UB}$, entonces el nodo puede ser acotado (optimalidad). Volver al Paso 2.
4. *Separación:* Decidir si un algoritmo de planos de corte debe ser ejecutado en N. Si la decisión es sí, entonces encontrar un conjunto de desigualdades válidas violadas por la solución $\bar{x}$ e insertarlas en M. Si se encontró alguna, volver al Paso 3.
5. *Branching:* Generar subproblemas $N_1, \cdots, N_k$ para el nodo actual N y meterlos en Q.

2.4.4. Algoritmos de Branch and price

En presencia de modelos MILP con un gran número de variables, incluir todas ellas explícitamente para ejecutar un algoritmo BC puede ser computacionalmente ineficiente o incluso imposible

en la práctica, por ejemplo, debido a limitaciones prácticas, como el espacio de memoria limitado. A modo de ejemplo, Savelsbergh [13] enfrentan este problema y consideran una solución conocida como *Branch and Price* (BP). La idea general de este algoritmo es combinar CG que permite resolver el problema LP para formulaciones grandes con el marco proporcionado por BB que incorpora restricciones de dominio entero. Básicamente, la formulación se restringe a un número limitado de columnas y, en cada nodo del árbol BB se ejecuta un algoritmo CG para resolver la relajación lineal.

La técnica BP puede ser mejorada aún más considerando un algoritmo de plano de corte para el problema maestro. En este caso, después de cada iteración CG, se puede ejecutar una rutina de separación. Estos tipos de algoritmos se llaman *Branch Cut and Price* (BCP). Sin embargo, la incorporación de planos de corte no es tan simple cuando se trabaja con un algoritmo CG para la relajación lineal. Recuerde que la estructura del problema maestro define el problema de pricing a resolver en cada iteración. En este sentido, agregar más restricciones, si no se hace con cuidado, puede introducir cambios en la estructura del problema de pricing y, por lo tanto, el algoritmo diseñado para el problema original puede requerir adaptaciones no triviales. Se dice que una familia de cortes es *robusta* si se puede utilizar el mismo algoritmo de pricing después de incorporarlos al modelo.

En los últimos años, el uso de los algoritmos BP y BCP ha aumentado dramáticamente. Hay dos razones detrás de esto. En primer lugar, las computadoras han mejorado a lo largo de los años, lo que lleva a más potencia de cálculo y capacidad de almacenamiento, lo que permite a los programadores manejar algoritmos y modelos más sofisticados. En segundo lugar, se ha realizado una extensa investigación sobre la generación de columnas y branch and price para comprender mejor los desafíos y oportunidades subyacentes de estas técnicas Desaulniers et al. [3].

2.5. El modelo de dependiente del tiempo

En esta tesis consideramos el modelo dependiente del tiempo introducido por Ichoua et al. [8]. En su modelo, consideran que las operaciones se llevan a cabo dentro de un horizonte de planificación $[0, T]$ que representa un día de trabajo típico. Luego, el horizonte se particiona en las llamadas zonas de velocidad $T_1 \cup T_2 \cup \cdots \cup T_k = [0, T]$ donde se asume que la velocidad es constante para cada arco. Posteriormente, se asigna un *perfil de velocidad* (o *función de velocidad*) $\nu_{ij}(t)$ a cada arco $(i, j) \in A$ que es una función escalonada con piezas constantes en cada zona de velocidad.

Dado que la velocidad ahora depende del tiempo, el tiempo de viaje de un arco específico $(i, j) \in A$ ya no es constante y depende del tiempo de salida t. Notar que aunque la velocidad es constante dentro de cada zona de velocidad, el vehículo puede cruzar sus límites mientras atraviesa el arco (i, j) y, por lo tanto, debe ajustar su velocidad en consecuencia.

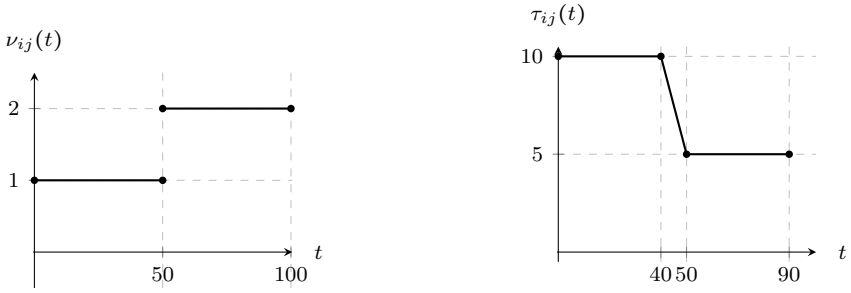
La función de tiempo de viaje $\tau_{ij}(t)$ indica el tiempo total de viaje del arco $(i, j) \in A$ al partir en el tiempo t. Sea v_{ijm} la velocidad constante del arco $(i, j) \in A$ en la zona de velocidad T_m para $m \in 1, \cdots, k$, entonces el Algoritmo 5 describe los pasos para calcular $\tau_{ij}(t)$. Vale la pena observar que la función τ_{ij} es continua y lineal a trozos. Además, satisface la propiedad First-In-First-Out (FIFO, por sus siglas en inglés), que indica que retrasar la salida no puede llevar a una llegada más temprana. Formalmente, $t + \tau_{ij}(t) \leq t' + \tau_{ij}(t')$ para todos $t < t'$ $(t, t' \in [0, T])$.

Como ejemplo, consideremos un arco $(i, j) \in A$ con una distancia $d_{ij} = 10$km, un horizonte de planificación definido por $T = 100$ minutos, y una función de velocidad $\nu_{ij}(t) = 1$ km/min para todo $t \in [0, 50]$ y $\nu_{ij}(t) = 2$ km/min para todo $t \in (50, 100]$. En este contexto, el tiempo de viaje del arco (i, j) al partir en $t_0 = 0$ es $\tau_{ij}(0) = 5$ minutos, ya que viaja todo el tiempo dentro de la primera zona de velocidad. Sin embargo, al partir en $t_1 = 45$, el tiempo de viaje se convierte en $\tau_{ij}(45) = 7{,}5$min ya que durante los primeros 5 minutos viaja a una velocidad de 1 km/min, y luego la velocidad se ajusta a 2 km/min requiriendo 2.5 minutos adicionales para completar los 5 km restantes. La Figura 2.3 muestra la función de velocidad y tiempo de viaje del arco de ejemplo (i, j).

A continuación, procedemos a definir una serie de funciones que representan diferentes aspectos dependientes del tiempo del problema o la solución. Definir estas funciones explícitamente simplifica

Algorithm 5 Algoritmo para calcular tiempos de viaje.

```
1: function TiempoViaje((i, j), t_0)
2:     Establecer m tal que t_0 ∈ T_m.
3:     t ← t_0,
4:     d ← d_ij,
5:     t' ← t + (d/v_ijm).
6:     while t' > máx(T_m) do
7:         d ← d − v_ijm(máx(T_m) − t),
8:         t ← máx(T_m),
9:         t' ← t + (d/v_ijm+1),
10:        m ← m + 1.
11:    devolver (t' − t_0).
```



(a) Función de velocidad $v_{ij}(t)$. *(b)* Función de tiempo de viaje $\tau_{ij}(t)$.

Fig. 2.3: Ejemplo numérico para la función de tiempo de viaje dependiente del tiempo.

la descripción y análisis de los algoritmos presentados en los siguientes capítulos.

2.5.1. Tiempo de finalización y duración

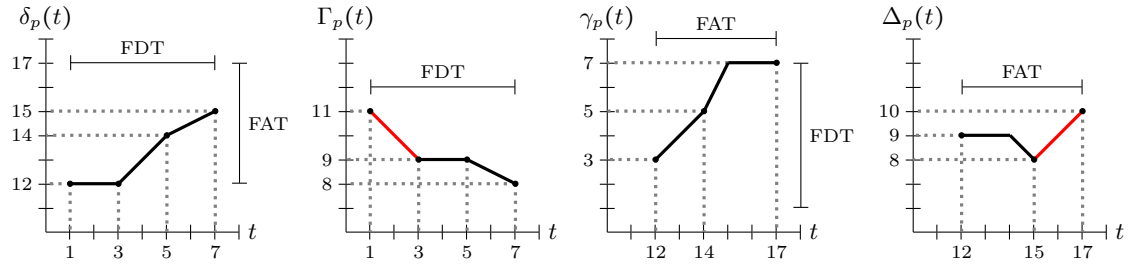
Sea $p = (v_1, \cdots, v_k)$ una ruta definida sobre el dígrafo $D = (V, A)$ con ventanas de tiempo $[a_i, b_i]$ y tiempo de servicio s_i para cada vértice $i \in V(D)$. Una pregunta interesante a considerar en un contexto dependiente del tiempo es cuándo es el tiempo más temprano en que se puede completar la ruta p (es decir, se visita el vértice v_k). La *función de tiempo de finalización* $\delta_p(t)$ indica el tiempo de finalización más temprano de la ruta p si se parte del vértice v_1 en el tiempo $t \in [0, T]$. Esta función se define recursivamente considerando las funciones de tiempo de viaje de los arcos intermedios. Sea $p^i = (v_1, \cdots, v_i)$ el prefijo de longitud $i-1$ de la ruta p, entonces $\delta_p = \delta_{p^k}$ se puede definir de la siguiente manera.

$$\delta_{p^i}(t) = \begin{cases} t & \text{si } i = 1 \\ \text{máx}\{a_{v_i}, \delta_{p^{i-1}}(t) + \tau_{v_{i-1}v_i}(\delta_{p^{i-1}}(t))\} + s_{v_i} & \text{en otro caso.} \end{cases}$$

Dado que todas las funciones de tiempo de viaje son lineales a trozos, se deduce que δ_p también es lineal a trozos. Intuitivamente, calcular el tiempo de finalización $\delta_p(t)$ al partir en algún momento t consiste en simular el viaje del vehículo a lo largo de la ruta p. La complejidad temporal es $\mathcal{O}(kM)$ donde M es el tamaño máximo de la función de tiempo de viaje para un arco $(i, j) \in p$.

Con respecto a la factibilidad de la ruta p, observar que cualquier vértice $v_i \in p$ debe ser visitado antes de su tiempo de cierre b_{v_i}. Esta restricción puede formalizarse utilizando la función de tiempo de finalización imponiendo que $\delta_{p^i}(0) \le b_{v_i}$ para cada $1 \le i \le k$. De hecho, dada una ruta p, podemos definir *tiempos de salida factibles* $\text{FDT}(p) = \{t \in [0, T] \mid \delta_{p^i}(t) \le b_{v_i} \text{ para cada } 1 \le i \le k\}$.

La Figura 2.4a muestra un ejemplo de una función de tiempo de finalización δ_p para alguna



(a) Tiempo de finalización δ *(b)* Duración Γ *(c)* Partida γ *(d)* Duración Δ

Fig. 2.4: Funciones genéricas de tiempo de llegada (a), duración en v_1 (b), tiempo de partida (c), y duración en v_k (d) para una ruta $p = (v_1, \ldots, v_k)$ con $[a_{v_1}, b_{v_1}] = [1, 7]$ y $[a_{v_k}, b_{v_k}] = [12, 17]$. En este ejemplo, se puede partir de v_1 en cualquier momento de sus ventanas de tiempo, por lo tanto $\mathrm{FDT}(p) = [1, 7]$. De manera similar, es posible esperar después de llegar a v_k en el tiempo 15, por lo tanto $\mathrm{FAT}(p) = [12, 17]$. Salir de v_1 antes del tiempo 3 produce una solución subóptima porque la visita a v_k no puede comenzar antes del tiempo 12. Esto se indica por la porción roja de Γ con pendiente -1. De manera similar, la pieza roja de Δ con pendiente 1 representa el aplazamiento de la visita a v_k en un tiempo $t > 15$. Notar que la duración mínima es $\Gamma(7) = \Delta(7) - 7 = 15 - 7 = 15 - \gamma(15) = \Delta(15)$, que corresponde a la salida de v_1 en el tiempo 7 y la llegada a v_k en el tiempo 15.

ruta p. Observar que es no decreciente, lo cual es consistente con el hecho de que las funciones de tiempo de viaje cumplen con la propiedad FIFO.

En algunos contextos, la función objetivo analiza la duración de las rutas en lugar del tiempo de finalización más temprano. La duración es el tiempo mínimo requerido para recorrer una ruta. Sea $p = (v_1, \cdots, v_k)$ una ruta, dado que $\delta_p(t)$ es el tiempo de finalización de servicio más temprano en el nodo v_k cuando p parte de v_1 en el tiempo $t \in \mathrm{FDT}(p)$, entonces $\Gamma_p(t) = \delta_p(t) - t$ es el tiempo mínimo requerido para recorrer p partiendo en el tiempo t. La función Γ_p, cuyo dominio es $\mathrm{FDT}(p)$, se conoce como la *duración de p en v_1* (Figura 2.4b). Como tanto δ_p como $t \to t$ son continuas y lineales a trozos, se deduce que Γ también es continua y lineal a trozos.

2.5.2. Definición Backward

La duración se define anteriormente desde el punto de vista de un recorrido hacia adelante de una ruta. En esta tesis, también necesitamos saber cuánto cuesta una ruta cuando su último vértice se visita en un tiempo t. Para mantener estas nociones juntas, definimos aquí la duración *backward* de una ruta.

Supongamos que $t_0 = a_{v_1}$ es un tiempo de partida factible para una ruta $p = (v_1, \ldots, v_k)$. Observar que v_k puede ser atendido inmediatamente en $\delta_p(t_0) \geq a_{v_k}$. Sin embargo, la visita a v_k también puede ser aplazada a cualquier tiempo $t \in [\delta_p(t_0), b_{v_k}]$ sin violar ninguna ventana de tiempo. Por lo tanto, el conjunto $\mathrm{FAT}(p) = [\delta_p(t_0), b_{v_k}]$ de *tiempos de llegada factibles* contiene $t \in [0, T]$ si y sólo si $t \leq b_{v_k}$ y cada vértice de p puede ser visitado dentro de su ventana de tiempo en un recorrido de p que visita v_k en el tiempo t. Claramente, $\mathrm{FAT}(p)$ está vacío o contiene b_{v_k} (Figura 2.4a). Para $t \in \mathrm{FAT}(p)$, el último tiempo en el que se puede partir de v_1 para visitar v_k en el tiempo t está dado por la *función de tiempo de partida* $\gamma_p : \mathrm{FAT}(p) \to \mathrm{FDT}(p)$ tal que $\gamma_p(t) = \max\{t' \in \mathrm{FDT}(p) \mid \delta_p(t') \leq t\}$ (Figura 2.4c). Siendo la contraparte de δ_p, no es difícil ver que γ_p también es continua, lineal a trozos y no decreciente. Notar que, por definición, $\Delta_p(t) = t - \gamma_p(t)$ es el tiempo mínimo requerido para recorrer p cuando v_k es visitado en el tiempo $t \in \mathrm{FAT}(p)$ (Figura 2.4d). El dominio de Δ_p es $\mathrm{FAT}(p)$ y, por uniformidad, nos referimos a Δ_p como la *duración de p en v_k*.

Por definición, p es factible si y sólo si $\mathrm{FAT}(p) \neq \varnothing$ y, en tal caso, su makespan (el mínimo tiempo en que es posible completarse el recorrido) es $\delta_p(a_{v_1}) = \min \mathrm{FAT}(p)$. Además, no es difícil ver que la duración c_p de p corresponde al mínimo entre las duraciones $\Delta_p(t)$ en v_k, es decir,

$c_p = \min\{\Delta_p(t) \mid t \in \mathrm{FAT}(p)\}$.

2.5.3. Propiedad de no espera

La inclusión de ventanas de tiempo introduce la posibilidad de tiempos de espera ociosos para los conductores siempre que lleguen a un cliente antes de su tiempo de apertura. Esta situación surge cuando las funciones de tiempo de viaje no satisfacen la propiedad de *no espera*. Formalmente, dado un arco $(i,j) \in A$, decimos que la función de tiempo de viaje $\tau_{ij}(t)$ satisface la propiedad de no espera si $t + \tau_{ij}(t) \geq a_j$ para cada $t \in [0,T]$. En otras palabras, si el tiempo de llegada a la ubicación $j \in V$ siempre es después del tiempo de apertura. Notar que siempre que las funciones de tiempo de viaje obedezcan esta propiedad, los tiempos de apertura pueden descartarse del problema, y la función de tiempo de finalización se puede simplificar a:

$$\delta_{p^i}(t) = \begin{cases} t & \text{si } i = 1 \\ \delta_{p^{i-1}}(t) + \tau_{v_{i-1}v_i}(\delta_{p^{i-1}}(t)) + s_{v_i} & \text{en otro caso.} \end{cases}$$

Las funciones objetivo clásicas consideradas para el TDTSP y variantes del TDVRPTW incluyen la duración total de las rutas o el tiempo de finalización más temprano basado en la función de tiempo de finalización δ. En estos escenarios, los tiempos de espera y los tiempos de viaje se tratan igualmente como un único recurso que es consumido por el vehículo mientras recorre la ruta. En este sentido, proponemos una nueva técnica de preprocesamiento que transforma las funciones de tiempo de viaje en una instancia para satisfacer la propiedad de no espera mientras se conserva el conjunto de rutas factibles y sus costos.

Supongamos que $\mathcal{I}$ es una instancia del TDVRPTW en la que la propiedad de no espera no se cumple para τ_{ij} para algunos $(i,j) \in A$. Entonces, la visita a j tiene que ser aplazada hasta a_j si (i,j) se recorre demasiado temprano. Definamos

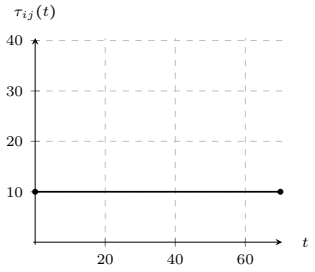
$$\hat{\tau}_{ij}(t) = \max\{a_j - t, \tau_{ij}(t)\} = \begin{cases} a_j - t & \text{si } \tau_{ij}(t) \leq a_j - t \\ \tau_{ij}(t) & \text{si } \tau_{ij}(t) > a_j - t \end{cases}$$

en el dominio $[0,T]$ para cada arco $(i,j) \in A$. Se puede ver fácilmente que cada función $\hat{\tau}_{ij}$ es una función de tiempo de viaje. De hecho, $\hat{\tau}_{ij}$ es lineal a trozos, continua y satisface la propiedad FIFO. Además, se deduce por inducción que la instancia $\hat{\mathcal{I}}$ obtenida al reemplazar τ_{ij} con $\hat{\tau}_{ij}$, para cada arco (i,j), es equivalente a $\mathcal{I}$.

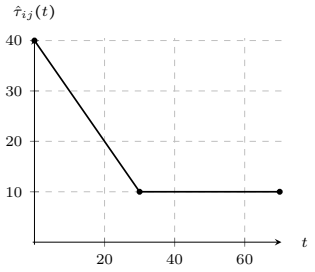
Proposición 2.1. *Dada una ruta $p = (v_1, \cdots, v_k = w)$ y un tiempo de partida $t \in [0,T]$, entonces el tiempo de finalización de la ruta p que parte en el tiempo t es el mismo en las instancias $\mathcal{I}$ y $\hat{\mathcal{I}}$ (es decir, $\delta_p(t) = \hat{\delta}_p(t)$).*

Demostración. Claramente, si $k = 1$, entonces $\delta_p(t) = \hat{\delta}_p(t)$. Por inducción, supongamos que la proposición se mantiene para cada camino con menos de k vértices. Entonces, $\delta_{p^{k-1}}(t) = \hat{\delta}_{p^{k-1}}(t) = t_1$. Por definición, $\delta_p(t) = \max\{a_w, t_1 + \tau_{v_{k-1},w}(t_1)\} + s_w$. Observemos que $\max\{a_w, t_1 + \tau_{v_{k-1},w}(t_1)\} = t_1 + \max\{a_w - t_1, \tau_{v_{k-1},w}(t_1)\} = t_1 + \hat{\tau}_{v_{k-1},w}(t_1)$. Por lo tanto, $\delta_p(t) = \max\{a_w, t_1 + \tau_{v_{k-1},w}(t_1)\} + s_w = t_1 + \hat{\tau}_{v_{k-1},w}(t_1) + s_w = \hat{\delta}_p(t)$. □

Ilustramos esto con un ejemplo sencillo que consta de sólo dos vértices, i y j, y donde la función de tiempo de viaje original se muestra en la Figura 2.5. En la instancia original, si el arco (i,j) se parte en el tiempo $t = 10$, entonces j se alcanza en el tiempo 20 y el conductor debe esperar hasta el tiempo $a_j = 40$. Sin embargo, en la instancia transformada el tiempo de viaje al partir de i en el tiempo $t = 10$ es 30, lo que significa que el vehículo llega exactamente en el tiempo de apertura del vértice j y no es necesario esperar.



(a) Función de tiempo de viaje τ_{ij} para la instancia $\mathcal{I}$.



(b) Función de tiempo de viaje $\hat{\tau}_{ij}$ para la instancia preprocesada $\hat{\mathcal{I}}$.

Fig. 2.5: Ejemplo numérico de la transformación de una función de tiempo de viaje de un solo período τ_{ij} en τ'_{ij} ($a_j = 40$, $T = 70$, $s_j = 0$)

2.5.4. Tiempos de servicio

Similar al caso de los tiempos de espera, una instancia del TDVRPTW puede ser transformada en una equivalente sin tiempos de servicio. Esto se puede hacer siempre y cuando los tiempos de servicio sean indistinguibles de los tiempos de viaje o espera, que es el caso cuando se optimiza el tiempo de finalización o la duración de las rutas. Nuevamente, realizar esta técnica de preprocesamiento simplifica aún más las funciones dependientes del tiempo, ya que, asumiendo que se cumple la propiedad de no espera, el tiempo de finalización ahora puede ser expresado como:

$$\delta_{p^i}(t) = \begin{cases} t & \text{si } i = 1 \\ \delta_{p^{i-1}}(t) + \tau_{v_{i-1}v_i}(\delta_{p^{i-1}}(t)) & \text{de lo contrario.} \end{cases}$$

Consideremos una instancia $\mathcal{I}$ del TDVRPTW en la que algún vértice $i \in V$ tiene un tiempo de servicio $s_i > 0$. Definamos $\hat{\tau}_{ij}(t) = s_i + \tau_{ij}(t)$ en el dominio $[0, T]$ para cada arco $(i, j) \in A$. Es fácil verificar que cada función $\hat{\tau}_{ij}$ es una función de tiempo de viaje. Además, $\hat{\tau}_{ij}$ es una función lineal a trozos, continua, y satisface la propiedad FIFO. Por inducción, la instancia $\hat{\mathcal{I}}$ obtenida al reemplazar τ_{ij} con $\hat{\tau}_{ij}$, para cada arco (i, j), es equivalente a $\mathcal{I}$.

Proposición 2.2. *Dada una ruta $p = (v_1, \cdots, v_k = w)$ y un tiempo de partida $t \in [0, T]$, entonces el tiempo de finalización de la ruta p que parte en el tiempo t es el mismo en las instancias $\mathcal{I}$ y $\hat{\mathcal{I}}$ (es decir, $\delta_p(t) = \hat{\delta}_p(t)$).*

Demostración. Para simplificar, asumimos que $\mathcal{I}$ satisface la propiedad de no espera, aunque, se puede realizar una prueba análoga para el otro caso. Claramente si $k = 1$, entonces $\delta_p(t) = \hat{\delta}_p(t)$. Por inducción, supongamos que la proposición se cumple para cada camino con menos de k vértices. Entonces, $\delta_p(t) = \delta_{p^{k-1}}(t) + \tau_{v_{k-1},w}(\delta_{p^{k-1}}(t)) + s_w = \hat{\delta}_{p^{k-1}}(t) + \tau_{v_{k-1},w}(\hat{\delta}_{p^{k-1}}(t)) + s_w = \hat{\delta}_{p^{k-1}}(t) + \hat{\tau}_{v_{k-1},w}(\hat{\delta}_{p^{k-1}}(t)) + 0 = \hat{\delta}_p(t)$. □

2.5.5. Funciones lineales a trozos

Cada función lineal a trozos continua, con un dominio acotado, puede representarse como una serie de puntos consecutivos (x_i, y_i) para $i = 1 \ldots |f|$, requiriendo $\Theta(n)$ espacio. Por ejemplo, la función con dominio $[-10, 10]$,

$$f(x) = \begin{cases} x & \text{si } x \geq 0 \\ -x & \text{si } x < 0 \end{cases}$$

puede representarse como una secuencia de tuplas $F = [(-10, 10), (0, 0), (10, 10)]$. Definimos la *longitud* de una función f, denotada por $|f|$, como su número de *piezas*. En este caso, $|f| = 2$. En cada función continua, vemos que el número de puntos de quiebre es $|f| + 1$. En este caso, los trozos

son $p_1 = ((-10,10),(0,0))$ y $p_2 = ((0,0),(10,10))$, y los puntos de quiebre son $b_1 = -10$, $b_2 = 0$, $b_3 = 10$.

Sin embargo, esta representación es apropiada sólo para funciones continuas. Cuando las funciones lineales a trozos no son continuas, entonces una alternativa es representarlas como una serie de segmentos $((x_i, y_i),(x_i', y_i'))$ para $i = 1 \cdots n$, ocupando el doble de memoria que la representación anterior. En este caso, el número de puntos de quiebre se mantiene entre $|f| + 1$ y $2|f|$. En los algoritmos presentados a lo largo de la tesis, el número de funciones almacenadas puede ser extremadamente grande, llegando eventualmente a ser una preocupación respecto al uso de memoria dentro de los algoritmos. Por lo tanto, elegir la representación adecuada para cada problema se convierte en una parte esencial del diseño del algoritmo.

Las operaciones clásicas sobre funciones lineales a trozos son sencillas y tienen la siguiente complejidad temporal.

Proposición 2.3. *Sea f y g dos funciones lineales, entonces, las operaciones de suma $(f + g)$, multiplicación $(f.g)$, composición $(f \circ g)$ y maximización* $\max\{f, g\}$ *toman tiempo* $\mathcal{O}(|f| + |g|)$.

Considerando la proposición anterior, podemos deducir una complejidad temporal para las funciones dependientes del tiempo de la Sección 2.5. Por ejemplo, dado un camino $p = (v_1, \cdots, v_k)$, denotemos por p^i al prefijo $(v_1, \cdots, v_i)$. Entonces, la función de tiempo de finalización $\delta_p = \delta_{p^k}$ puede escribirse como una combinación de las operaciones de la Proposición 2.3 de la siguiente manera:

$$\delta_{p^i} = (t \to t) + (\tau_{v_{i-1}, v_i} \circ \delta_{p^{i-1}}),$$

donde $(t \to t)$ es la función identidad. Por lo tanto, la complejidad de calcular δ_p es $\mathcal{O}(M^2 k)$ con M la longitud de la función de tiempo de viaje más larga.

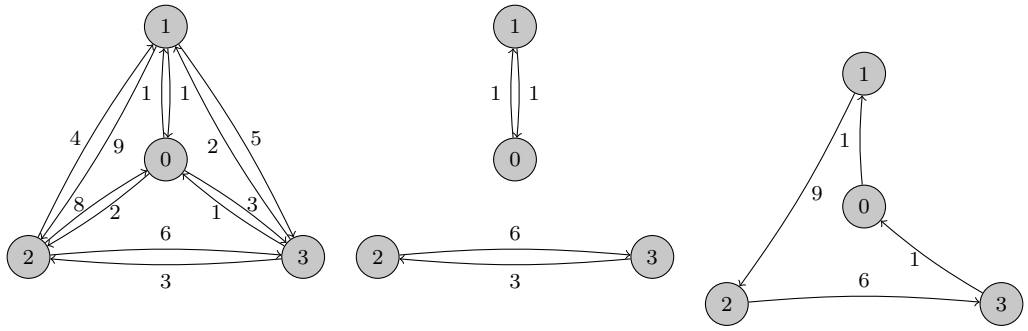
Se pueden idear ecuaciones análogas para el resto de las funciones dependientes del tiempo como γ_p, Γ_p y Δ_p. Por simplicidad, omitimos los detalles.

2.6. Caso de estudio 1: El Problema del Viajante de Comercio

Este ejemplo ofrece una visión general de la metodología aplicada para resolver problemas de optimización utilizando modelos MILP. La idea es proponer una formulación MILP sencilla que represente el Problema del Viajante de Comercio (Asimétrico), la cual puede ser resuelta posteriormente usando cualquier solver MILP de propósito general.

Recordemos que el TSP consiste en un digrafo completo $D = (V, A)$ donde cada vértice $i \in V$ representa una ubicación y cada par de vértices $i, j \in V$ $(i \neq j)$ está conectado mediante un arco $(i, j) \in A$ que tiene un costo positivo c_{ij}. Una solución al TSP es un ciclo Hamiltoniano de costo mínimo, es decir, un ciclo que visita todos los vértices exactamente una vez de modo que se minimice la suma de los costos de los arcos. La Figura 2.6a muestra una instancia del TSP con cuatro vértices y cada arco tiene un número asociado que representa su costo.

El primer paso para construir una formulación MILP es definir la semántica de las variables. Para cada arco $(i, j) \in A$, se define la variable binaria x_{ij} que indica si el arco $(i, j) \in A$ es considerado en la solución. En otras palabras, $x_{ij} = 1$ si y sólo si el arco $(i, j) \in A$ pertenece a la solución óptima. Luego, procedemos a definir el resto de la formulación MILP que incluye la función objetivo y las restricciones. Para simplificar la notación, definimos el conjunto de arcos salientes (entrantes) de un conjunto de vértices $S \subseteq V$ como $N^+(S) = \{(i, j) \in A \mid i \in S\}$ $(N^-(S) = \{(i, j) \in A \mid j \in S\})$.



(a) Instancia del TSP con cuatro vértices. *(b)* Solución infactible (subrutas). Costo: 11. *(c)* Solución óptima. Costo: 12.

Fig. 2.6: Ejemplo del TSP Asimétrico.

$$\text{mín} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.5}$$

s.t.

$$\sum_{(i,j)\in N^+(\{i\})} x_{ij} = 1 \qquad \forall i \in V, \tag{2.6}$$

$$\sum_{(i,j)\in N^-(\{i\})} x_{ij} = 1 \qquad \forall i \in V, \tag{2.7}$$

$$\sum_{(i,j)\in N^+(S)} x_{ij} \geq 1 \qquad \forall \varnothing \subset S \subset V, \tag{2.8}$$

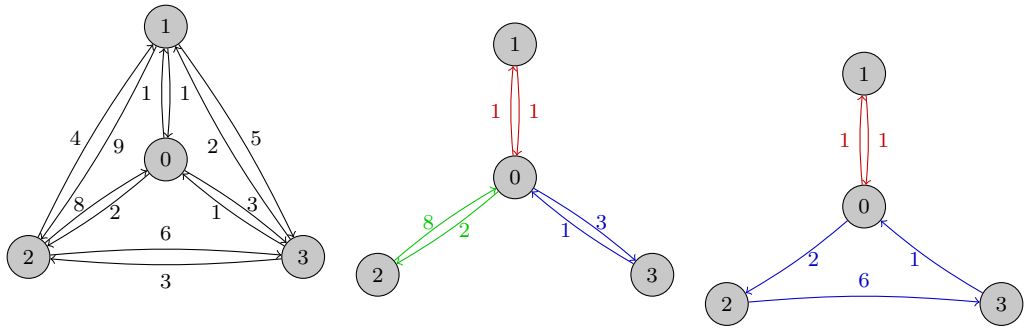
$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{2.9}$$

La función objetivo (2.5) minimiza la suma de los costos de los arcos seleccionados. Las restricciones (2.6)-(2.7) imponen que cada vértice debe tener exactamente un arco entrante y un arco saliente seleccionados, ya que estas condiciones son satisfechas por cualquier vértice dentro de un ciclo (Hamiltoniano). Sin embargo, notar que estas condiciones no son suficientes. La Figura 2.6b ilustra un escenario en el que las restricciones (2.6)-(2.7) y (2.9) se cumplen, pero, no obstante, la solución no representa un ciclo Hamiltoniano. La razón es que contiene los llamados *subtours* (ciclos con menos de $|V|$ vértices). Para evitar tales soluciones inviables, las restricciones (2.8), conocidas como las *Restricciones de Eliminación de Subtours* (SEC, por sus siglas en inglés), imponen que en cualquier subconjunto propio de los vértices, al menos un arco debe salir hacia un vértice externo. La solución en la Figura 2.6b es inviable ya que viola las SEC para $S = \{0,1\}$. Observar que las restricciones (2.8) definen un número exponencial de desigualdades y por lo tanto no pueden ser incluidas explícitamente en la formulación. Típicamente, esto se maneja incorporando estas desigualdades bajo demanda en un algoritmo BC, dado que la rutina de separación de las SEC es polinomial. Finalmente, presentamos una solución óptima (factible) en la Figura 2.6c que satisface todas las restricciones y tiene el costo mínimo.

2.7. Caso de estudio 2: El Problema de Ruteo de Vehículos con Capacidades

Esta sección ofrece un ejemplo de los diferentes aspectos que deben considerarse al diseñar un algoritmo Branch-and-Price.

El Problema de Ruteo de Vehículos con Capacidades (CVRP, por sus siglas en inglés) considera un digrafo completo $D = (V, A)$ donde los vértices $V = V_c \cup \{0\}$ representan a los clientes $V_c = \{1, \ldots, n\}$ y un vértice distinguido $0 \in V$ representa el depósito donde todas las rutas deben



(a) Instancia del CVRP con cuatro vértices. *(b)* Solución inicial (loops). Costo: 16. *(c)* Solución óptima. Costo: 11.

Fig. 2.7: Ejemplo del CVRP.

comenzar y terminar. Cada arco $(i, j) \in A$ conecta un par de ubicaciones y tiene un costo asociado $c_{ij} \geq 0$ que se paga si se recorre el arco. Además, existe una flota ilimitada de vehículos homogéneos con capacidad Q y cada cliente $i \in V_c$ tiene una demanda $q_i \geq 0$ asociada, medida en la misma unidad. Para simplificar, considere $q_0 = 0$. Una solución para el CVRP consiste en un conjunto de rutas que deben satisfacer (i) cada ruta comienza y termina en el depósito, (ii) la suma de las demandas de los clientes de cada ruta no excede la capacidad del vehículo Q, y (iii) cada cliente es visitado exactamente una vez. El objetivo es encontrar la solución con el costo total mínimo.

De manera similar al ejemplo anterior, comenzamos definiendo la semántica de las variables. Sea Ω el conjunto de rutas factibles, es decir, que satisfacen (i) y (ii). Para cada ruta $r \in \Omega$, sea la variable binaria $y_r \in \{0, 1\}$ que indica si la ruta r pertenece a la solución óptima. Además, sea c_r el costo de recorrer todos los arcos en la ruta r, y para cada vértice $i \in V$ defina la constante a_{ir} que toma el valor 1 si el vértice i es visitado en r, y 0 en caso contrario. Entonces, la siguiente formulación MILP conocida como la *formulación de partición de conjuntos* (SPF, por sus siglas en inglés) modela el CVRP:

$$\text{mín} \sum_{r \in \Omega} c_r y_r \tag{2.10}$$

$$s.a.$$

$$\sum_{r \in \Omega} a_{ir} y_r = 1 \qquad \forall i \in V_c, \tag{2.11}$$

$$y_r \in \{0, 1\} \qquad \forall r \in \Omega. \tag{2.12}$$

La función objetivo (2.10) minimiza los costos totales de las rutas seleccionadas. Las *restricciones de grado* (2.11) imponen que para cada cliente $i \in V_c$ debe haber exactamente una ruta activa que lo visite. Finalmente, las restricciones (2.12) aseguran que las variables y sean binarias.

La Figura 2.7a muestra un ejemplo del CVRP con $n = 3$ clientes con demanda $q_i = 10$ para todo $i \in V_c$ y una capacidad del vehículo $Q = 20$. Luego, la Figura 2.7b ilustra una solución inicial que corresponde a los llamados *loops* donde cada cliente es visitado en una ruta diferente. Observar que esta solución es factible en todas las instancias. Finalmente, la Figura 2.7c presenta una solución óptima que consiste en las rutas $r_1 = (0, 1, 2, 0)$ y $r_2 = (0, 3, 0)$. Por lo tanto, la valoración óptima $\bar{y}$ del modelo anterior es $\bar{y}_{r_1} = \bar{y}_{r_2} = 1$ y $\bar{y}_{r'} = 0$ para todo $r' \in \Omega \setminus \{r_1, r_2\}$.

A diferencia del caso del TSP, el número de rutas factibles en una instancia general de CVRP es exponencial, y por lo tanto la SPF no puede definirse explícitamente en la práctica. Recuerde que las formulaciones con un gran número de variables pueden manejarse con algoritmos de CG y BP. El problema dual asociado con la SPF (2.10)-(2.12) se describe a continuación:

$$\text{máx} \sum_{i \in V_c} \pi_i \tag{2.13}$$

s.t.

$$\sum_{i \in V_c} a_{ir} \pi_i \leq c_r \qquad \forall r \in \Omega. \tag{2.14}$$

Observar que el problema dual incluye una restricción para cada variable (o ruta) y una variable para cada restricción (o cliente) de la formulación primal. En un marco de CG, comenzamos con un Problema Maestro Restringido (RMP, por sus siglas en inglés) con un conjunto limitado de rutas $\Omega' \subseteq \Omega$, por ejemplo considerando $r_i = (0, i, 0)$ para $i \in V_c$ representado en la Figura 2.7b. Luego, después de reoptimizar el LP, obtenemos valores duales óptimos para el RMP y resolvemos el problema de pricing. En este caso, debemos encontrar una restricción (2.14) que no sea satisfecha por $\bar{\pi}$, es decir, encontrar una ruta $r \in \Omega \setminus \Omega'$ tal que $c_r - \sum_{i \in V_c} a_{ir} \pi_i < 0$. Dada una ruta $r \in \Omega$, definimos su costo reducido $\bar{c}_r = c_r - \sum_{i \in V_c} a_{ir} \pi_i$. Entonces, podemos definir el problema de pricing como encontrar una ruta (factible) con un costo reducido negativo.

Antes de proceder a la solución del problema de pricing, notar que el SPF es ajeno a las restricciones operativas que definen si una ruta es factible o no. De hecho, esto está codificado en la definición de Ω, ya que una variable y_r solo se crea para las rutas $r \in \Omega$, que son factibles por definición. Como resultado, este método es flexible con respecto a las restricciones operativas intra-ruta tales como demandas, ventanas de tiempo y recolecciones y entregas. La diferencia radica únicamente en el algoritmo de pricing que se utilizará para cada variante específica.

En el SPF para el CVRP, el problema de pricing se puede plantear como encontrar una ruta r que comience y termine en el depósito, visite a cada cliente a lo sumo una vez, y tenga un costo reducido negativo $\bar{c}_r < 0$. Observar que si para cada arco $(i, j) \in A$ redefinimos el costo del arco como $c^*_{ij} = c_{ij} - \pi_j$, entonces el problema se reduce a encontrar el camino elemental más corto (factible) que comience y termine en el depósito considerando los pesos de los arcos c^*. Este problema es conocido como el Problema de Camino Mínimo Elemental con Restricciones de Recursos (ESPPRC, por sus siglas en inglés).

Los métodos de última generación típicamente utilizan enfoques basados en programación dinámica, comúnmente conocidos como *algoritmos de labeling*, para resolver el ESPPRC. La idea es generar un árbol de enumeración que contenga todos los caminos factibles que parten del depósito de manera constructiva. Cada nodo de este árbol se denomina *label* y almacena información sobre el camino parcial correspondiente. En este caso, cada camino $p = (0, w_1, \cdots, w_i = v)$ está representado por un label $L = (prev, v, q, c, S)$ donde $prev(L)$ es un puntero al label que representa el subcamino anterior, $v(L)$ es el último vértice de p, $q(L)$ es la suma de las demandas de los clientes, $c(L)$ el costo reducido del camino, y $S(L)$ el conjunto de clientes visitados.

El Algoritmo 6 muestra una visión general de los pasos básicos de un método de labeling. El algoritmo comienza con un label inicial $L_0 = (\perp, 0, 0, 0, \varnothing)$ en una cola. Luego, cada label L se extiende a todos los vértices $w \in V$, y la extensión se considera factible si (i) $w \notin S(L)$ y (ii) $q_w + q(L) \leq Q$. Si la extensión no es una ruta completa, se añade a la cola. Finalmente, para reducir aún más el tamaño del árbol de enumeración, consideramos las llamadas *reglas de dominancia* que permiten descartar de forma segura *labels* (es decir, caminos parciales) para las cuales cada posible extensión es peor en comparación con otro camino ya explorado. Dados dos labels M y L, decimos que M domina a L si, para toda extensión factible E de L que forme una ruta completa $r = L + E$, la ruta $r' = M + E$ también es factible y satisface $c_{r'} \leq c_r$. Dado que esta regla es computacionalmente difícil de verificar, consideramos en su lugar las siguientes condiciones suficientes.

Regla 2.1 (Regla de Dominancia)**.** *Un label M domina a otro label L cuando $v(M) = v(L)$, $q(M) \leq q(L)$, $c(M) \leq c(L)$, y $S(M) \subseteq S(L)$.*

Intuitivamente, estas condiciones aseguran que M representa un estado en el mismo nodo con carga o costo no mayores, y habiendo visitado un subconjunto de los nodos visitados por L, lo que garantiza que cualquier extensión factible desde L también es factible desde M.

Una vez que todas las etiquetas han sido extendidas y procesadas, el algoritmo devuelve la ruta factible con el costo reducido mínimo que ha sido enumerada. Si el costo es negativo, se añade al RMP, de lo contrario el algoritmo CG termina.

Algorithm 6 Visión general de un algoritmo de labeling.

Entrada: Una instancia del ESPP.

1. *Inicialización:* Crear una cola $\mathcal{Q}$ con un label inicial $(\perp, 0, 0, 0, \varnothing)$, y una estructura procesada $\mathcal{P}$ donde se almacenan los labels no dominados.
2. *Enumeración:* Si Q está vacía, retornar la mejor solución enumerada. De lo contrario, extraer el primer label L.
3. *Dominancia:* Si algún label en $\mathcal{P}$ domina a L, entonces descartar L y regresar al Paso 2.
4. *Extensión:* Para cada vértice $w \in V$, crear el label L_w como la extensión del label L a través del arco $(v(L), w)$. Si L_w es inviable, regresar al Paso 2.
5. Si $v(L_w) = 0$ entonces L_w representa una solución, conservarla si es la mejor solución encontrada hasta ahora.
6. Si $v(L_w) \neq 0$ entonces insertar L_w en $\mathcal{Q}$, actualizar $\mathcal{P}$ con L_w, y regresar al Paso 2.

Referencias

[1] Vasek Chvátal, ed. *Linear Programming*. W. H. Freeman, 1983.

[2] Harlan Crowder et al. «Solving Large-Scale Zero-One Linear Programming Problems». En: *Operations Research* 31.5 (1983), págs. 803-834.

[3] Guy Desaulniers et al., eds. *A Primer in Column Generation*. Boston, MA: Springer US, 2005. ISBN: 978-0-387-25486-9.

[4] Michael Drexl et al. «Solving elementary shortest-path problems as mixed-integer programs». En: *OR Spectrum* 36.2 (2014), págs. 281-296. ISSN: 14366304.

[5] Michael R. Garey et al. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990. ISBN: 0716710455.

[6] Ralph E. Gomory. «Outline of an algorithm for integer solutions to linear programs». En: *Bull. Amer. Math. Soc.* 64.5 (1958), págs. 275-278.

[7] Martin Grötschel et al. «A Cutting Plane Algorithm for the Linear Ordering Problem». En: *Operations Research* 32 (1984), págs. 1195-1220.

[8] Soumia Ichoua et al. «Vehicle dispatching with time-dependent travel times». En: *Eur. J. Oper. Res.* 144.2 (2003), págs. 379-396.

[9] N Karmarkar. «A new polynomial-time algorithm for linear programming». En: *Combinatorica* 4 (1984), págs. 373-395.

[10] George L. Nemhauser et al. *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience, 1988. ISBN: 0-471-82819-X.

[11] Manfred. Padberg et al. «A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems». En: *SIAM Review* 33.1 (1991), págs. 60-100.

[12] Steffen Rebennack. «Stable set problem: branch & cut algorithmsStable Set Problem: Branch & Cut Algorithms». En: *Encyclopedia of Optimization*. Ed. por Christodoulos A. Floudas et al. Boston, MA: Springer US, 2009, págs. 3676-3688. ISBN: 978-0-387-74759-0.

[13] Martin Savelsbergh. «A Branch-and-Price Algorithm for the Generalized Assignment Problem». En: *Operations Research* 45.6 (1997), págs. 831-841.

3. UN ALGORITMO BRANCH AND CUT PARA EL PROBLEMA DE CICLOS CON BENEFICIOS DEPENDIENTES DEL TIEMPO CON RESTRICCIONES DE RECURSOS

3.1. Introducción

En los últimos años ha habido una tendencia a enriquecer los modelos clásicos para el VRP incorporando funciones de tiempo de viaje más complejas para capturar el efecto de la congestión, conocidos como Problemas de Ruteo de Vehículos con Tiempos de Viaje Variables (TDVRP, por sus siglas en inglés, véase, por ejemplo, Gendreau et al. [11]). Aunque se han propuesto diferentes modelos para capturar este efecto en los tiempos de viaje, el propuesto en Ichoua et al. [13] se ha convertido en un estándar dentro de la literatura sobre TDVRP.

Los métodos exactos para variantes multi-vehículo del VRP clásico generalmente recurren a técnicas MILP y de descomposición, resultando en algoritmos BP y BCP. El problema del profitable tour con tiempos variables y restricciones de recursos (TDPTPRC, por sus siglas en inglés) surge de forma natural como el problema de pricing dentro de algoritmos de generación de columnas para enfoques basados en partición de conjuntos para TDVRPs multi-vehículo. En general, la presencia de los llamados recursos codifica restricciones operativas intra-ruta y los beneficios representan las variables duales asociadas a las restricciones vinculadas a los vértices en el problema maestro. En su versión independiente del tiempo, el TDPTPRC puede formularse como un Problema de Camino Mínimo Elemental con Restricciones de Recursos (ESPPRC, por sus siglas en inglés), que se sabe es fuertemente $\mathcal{NP}$-Hard. Desde un punto de vista algorítmico, si uno de los recursos se conoce como *restrictivo* y (potencialmente) limita la longitud de las rutas, el problema de pricing se aborda usando algoritmos de labeling (véase, por ejemplo, Feillet [10]). Esta es la estrategia seguida por Dabia et al. [7] para el TDVRP con ventanas de tiempo (TDVRPTW, por sus siglas en inglés); Sun et al. [20] para el TDPTP con capacidad de vehículo, ventanas de tiempo y recolecciones y entregas; Sun et al. [22] para el TDVRP con recolecciones y entregas; y Huang et al. [12] para una variante del TDVRPTW con flexibilidad de ruta.

Si no se dispone de tal recurso, una alternativa es considerar una relajación del problema permitiendo algún tipo de inviabilidad, que luego se maneja durante la enumeración, a costa de deteriorar eventualmente la calidad del límite inferior de la relajación LP. El estándar actual y el enfoque más efectivo es la relajación *ng-route* propuesta por Baldacci et al. [5] para el VRP, y utilizada en el contexto de los TDVRPs por Spliet et al. [19] para una variante del problema de asignación de ventanas de tiempo con tiempos de viaje dependientes del tiempo. En el contexto del VRP clásico, otra línea de investigación considera abordar el ESPPRC usando programación entera mixta, como se propone en Drexl et al. [9], Drexl [8], Jepsen et al. [14] y Taccari [23]. Destacamos los dos últimos artículos, donde se desarrollan algoritmos BC. Una de las principales motivaciones para explorar esta técnica radica en la flexibilidad para incorporar diferentes restricciones operativas así como estructuras particulares inducidas por branching y/o cortes incluidos en el problema maestro.

Los problemas de ruteo de vehículos con beneficios, incluyendo el PTP y el TDPTPRC, tienen su propia relevancia práctica en diferentes contextos, como se indica en Toth et al. [24, Capítulo 10]. En este tipo de problemas, el conjunto de clientes a atender no se da de antemano y se convierte en parte de la decisión a tomar además del orden en que deben ser visitados. La mayor parte de la investigación en este aspecto se ha dedicado al caso de un solo vehículo, incluyendo el *problema del profitable tour* (PTP, por sus siglas en inglés), el *problema del viajante con recolección de premios* (PCTSP, por sus siglas en inglés) y el *problema de orienteering* (OP, por sus siglas en inglés), entre otros (véase, por ejemplo, Toth et al. [24] para una revisión actualizada de la literatura). Específicamente para los TDVRPs, el TDPTP con ventanas de tiempo y recolecciones y entregas ha sido abordado con una formulación MILP que explota los llamados *travel-time breakpoints*, que denominamos TTBF, y mediante algoritmos de labeling en Sun et al. [20]. Una formulación

MILP similar y diferentes heurísticas han sido propuestas para el TDOP en Verbeeck et al. [25] y posteriormente para el TDOP con ventanas de tiempo en Verbeeck et al. [26].

Una motivación adicional para estudiar los TDVRP con beneficios es que los hallazgos y desarrollos pueden extenderse a variantes más clásicas de un solo vehículo, como el problema del viajante de comercio con tiempos de viaje variables (TDTSP, por sus siglas en inglés) y el TDTSP con ventanas de tiempo (TDTSPTW, por sus siglas en inglés), donde se requiere visitar a todos los clientes. Estas dos variantes han recibido cierta atención recientemente, en particular respecto a algoritmos exactos. El TDTSP se considera primero en Cordeau et al. [6], donde se estudia la función de tiempo de viaje dependiente del tiempo y los autores proponen una cota inferior obtenida al resolver un TSP auxiliar independiente del tiempo. Se demuestra que la cota es ajustada bajo ciertas condiciones de tráfico, y posteriormente se incorpora en un algoritmo BC. Recientemente, este enfoque se mejora en Arigliano et al. [2]. Una extensión del marco propuesto en Cordeau et al. [6] se considera para el TDTSPTW en Arigliano et al. [3]. Desde un punto de vista computacional, la efectividad del enfoque depende en gran medida de la calidad de la cota inferior inicial, que el algoritmo BC encuentra difícil de mejorar durante la enumeración en el árbol BC. En Montero et al. [17] se propone un algoritmo BC más clásico, donde se estudia una formulación MILP propuesta en Sun et al. [20] y se fortalece con planos de corte clásicos, preprocesamiento y varias heurísticas iniciales generalizadas del TSP con ventanas de tiempo. Los resultados superan a los reportados en Arigliano et al. [3]. Recientemente, en Vu et al. [27] se ha propuesto un algoritmo de descubrimiento de discretización dinámica (DDD) para el TDTSPTW. Hasta donde sabemos, al momento de la publicación del artículo del cual se origina este capítulo, este era el algoritmo más efectivo disponible en la literatura para el TDTSPTW. A diferencia de enfoques previos que se centraban únicamente en minimizar el makespan, puede adaptarse para minimizar la duración total del tour demorando opcionalmente la salida desde el depósito.

En este sentido, desarrollamos un algoritmo BC para el TDPTPRC basándonos en la investigación previa presentada en Montero et al. [17] y Sun et al. [20]. Las principales contribuciones de *este capítulo* son las siguientes. Primero, proponemos una formulación adaptada para el TDPTPRC que, dependiendo de la estructura de la función de tiempo de viaje dependiente del tiempo, reduce el número total de variables y restricciones. Además, demostramos que esta adaptación no tiene un impacto en términos de la calidad de la cota inferior obtenida por la formulación LP. Segundo, derivamos varias familias de desigualdades válidas para este modelo, enfocándonos en explotar la estructura dependiente del tiempo para el problema. Específicamente, proponemos cuatro familias de desigualdades válidas, incluyendo una que generaliza una familia bien conocida de problemas relacionados. Tercero, desarrollamos rutinas eficientes de separación para las nuevas familias de desigualdades válidas que, combinadas con otras desigualdades válidas de la literatura, se incorporan en un algoritmo BC adaptado. Cuarto, realizamos extensos experimentos computacionales para evaluar el algoritmo BC en instancias de referencia de la literatura para cuatro problemas diferentes y distintas funciones objetivo y restricciones, dando un total de 6 contextos dependientes del tiempo diferentes. En particular, consideramos una variante del TDPTPRC en instancias de pricing para el TDVRP sin ventanas de tiempo, donde la capacidad no es un recurso limitante, obtenida a partir de las conocidas instancias de Solomon. También adaptamos el modelo al TDOP sin ventanas de tiempo. Adicionalmente, incorporamos ventanas de tiempo y recolecciones y entregas al TDPTP y comparamos con los resultados reportados para una formulación MILP estándar en Sun et al. [20], los cuales se mejoran significativamente. Finalmente, adaptamos la formulación y el algoritmo BC al TDTSPTW y evaluamos la eficiencia utilizando los benchmarks propuestos en Vu et al. [27], mostrando que nuestro enfoque es capaz de mejorar los resultados presentados en dos de los cuatro escenarios considerados.

El resto del capítulo está organizado de la siguiente manera. En la Sección 3.2 definimos el TDPTPRC y presentamos la mayor parte de la notación utilizada a lo largo del capítulo. Una nueva formulación MILP compacta se presenta y describe en la Sección 3.3, incluyendo resultados teóricos sobre la calidad de la relajación LP. Las desigualdades válidas se presentan en la Sección 3.4, y los detalles respecto al algoritmo BC se discuten en la Sección 3.5. Los experimentos numéricos se reportan en la Sección 3.6 y, finalmente, presentamos las conclusiones y líneas futuras en la

Sección 3.7.

3.2. Definición del problema

En esta sección, definimos el TDPTPRC en su forma general. Remitimos al lector a Dabia et al. [7] y Lera-Romero et al. [15] para detalles sobre su incorporación dentro de un framework de BP basado en una formulación de partición de conjuntos para el TDVRP.

La red con tiempos de viaje variables se define de la siguiente manera. Sea $D = (V, A)$ un digrafo, con $V = \{0 = v_s, 1, 2, \cdots, n, v_e = n+1\}$ el conjunto de vértices que representan a los clientes, y A el conjunto de arcos que representan sus conexiones. Dos vértices distinguidos, v_s y v_e, denotan el punto de inicio y final del camino, respectivamente. Típicamente, estos dos vértices representan el depósito y por lo tanto no se consideran arcos entrantes a v_s ni arcos salientes desde v_e.

Generalmente, los recursos utilizados para modelar restricciones de la vida real son la capacidad del vehículo, las ventanas de tiempo, los tiempos de servicio y las recolecciones y entregas, entre otros. En la versión con capacidades, los vehículos tienen una capacidad fija Q y cada cliente $i \in V$ tiene una demanda q_i, medida en la misma unidad. La suma de las demandas de los clientes en una sola ruta de vehículo no debe exceder su capacidad. Otra restricción son las llamadas ventanas de tiempo, que se definen mediante un intervalo de tiempo $[a_i, b_i]$ para cada $i \in V$, indicando el intervalo de tiempo en el que es factible atender al cliente i. En algunos contextos, el tiempo de servicio s_i para cada cliente $i \in V$ es considerable, y por lo tanto decimos que el vehículo permanece al menos s_i unidades de tiempo en el cliente i prestándole servicio. Nótese que bajo la presencia de ventanas de tiempo, el servicio debe comenzar dentro de sus límites. En cuanto a las variantes de orienteering, denotamos por $t_{\max}$ la duración máxima del recorrido. Finalmente, algunos problemas involucran cumplir un conjunto de solicitudes R en lugar de clientes, donde cada solicitud $r_k \in R$ tiene un cliente k del cual se deben recoger productos y un cliente, generalmente notado como $n+k$, al que deben ser transportados los productos recogidos en k.

La dependencia temporal se modela utilizando el modelo de tiempo de viaje propuesto en Ichoua et al. [13] y explicado en detalle en la Sección 2.5. Recuerde que las operaciones se realizan dentro de un horizonte temporal $[0, T]$, y cada arco $(i, j) \in A$ tiene asociada una función de tiempo de viaje $\tau_{ij}(t)$ que indica el tiempo requerido para llegar a j desde i si se parte en el tiempo $t \in [0, T]$. Sea $T^{ij} = \{T_1^{ij}, \ldots, T_M^{ij}\}$ una partición del horizonte de planificación en los llamados *periodos de tiempo de viaje* para el arco $(i, j) \in A$, donde $T_m^{ij} = [w_{ij}^m, w_{ij}^{m+1}]$. Los valores w_{ij}^m se denominan puntos de quiebre del tiempo de viaje. Dentro de cada periodo de tiempo de viaje, se asume que la función de tiempo de viaje es conocida y lineal. Denotamos con $\tau_{ij}(t) = \theta_{ij}^m t + \eta_{ij}^m$, para $t \in T_m^{ij}$, a la función lineal de tiempo de viaje en el periodo de tiempo $m \in T^{ij}$.

Se consideran diferentes funciones objetivo para este problema. Consideramos una variante donde cada cliente $i \in V$ tiene un beneficio π_i que se obtiene si el cliente i es visitado, y por lo tanto, dada una ruta $p = (v_s, v_1, \ldots, v_k, v_e)$ se optimiza un compromiso entre su costo c_P y el beneficio $\sum_{v \in p} \pi_v$. El costo c_p de una ruta —recordando la notación introducida en el Capítulo 2— puede ser su makespan ($\delta_p(0)$), que es el tiempo más temprano en que puede finalizar; su duración ($\min_{t \in \mathrm{FAT}(p)} \Delta(t)$), siendo el tiempo mínimo requerido para recorrer la ruta eventualmente demorando la partida y no necesariamente comenzando en $t = 0$; o cualquier otra métrica. El TDPTPRC implica encontrar un camino elemental $p = (v_s, v_1, \ldots, v_k, v_e)$ que minimice $c_p - \sum_{v \in p} \pi_v$, comenzando y terminando dentro del horizonte de planificación cumpliendo todas las restricciones de recursos.

Finalmente, introducimos alguna notación adicional que se ha utilizado a lo largo del capítulo. Sea $N^-(j) = \{(i, j) \in A\}$, $N^+(i) = \{(i, j) \in A\}$, $N^+(S, T) = \{(i, j) \in A : i \in S,\ j \in T\}$, $\bar{S} = V - S$. Finalmente, nos referimos a la combinación de un arco (i, j) y un período de tiempo $m \in T^{ij}$ como el arco (dependiente del tiempo) (i, j, m), y llamamos $\hat{A} = \{(i, j, m) \in A \times \mathbb{N} : (i, j) \in A, m \in T^{ij}\}$ al conjunto de todos ellos.

Asumimos en las secciones restantes de este capítulo que las instancias no contienen tiempos de servicio, y también que no requieren tiempos de espera porque si los requieren, podemos transformarlas previamente y trabajar con las nuevas instancias en su lugar (ver secciones 2.5.3 y 2.5.4).

Usamos estas propiedades en la Sección 3.4 para definir nuevas familias de desigualdades válidas.

3.3. Formulaciones ILP

3.3.1. Modelo 1: TTBF

A continuación presentamos una formulación derivada de la presentada en Sun et al. [21] y considerada también en Montero et al. [17]. Este modelo se denomina Formulación Travel-Time Breakpoints (TTBF, por sus siglas en inglés). Las variables binarias x_{ij}^m indican que la ruta viaja de i a j, partiendo de i dentro del período de tiempo de viaje $m \in T^{ij}$. Definimos las variables binarias tradicionales x_{ij} en términos de x_{ij}^m. La variable binaria y_i toma el valor 1 si y sólo si el vértice $i \in V$ es visitado. Además, para cada vértice una variable no negativa t_i indica el tiempo de partida desde $i \in V$, si es visitado por la ruta. El valor de t_i se descompone para considerar el tiempo de partida para cada arco específico y período de tiempo de viaje, si se selecciona alguno. Por lo tanto, definimos las variables $t_{ij}^m = t_i$ si $x_{ij}^m = 1$, y $t_{ij}^m = 0$ en caso contrario.

$$\text{mín} \quad t_{v_e} - t_{v_s} - \sum_{i \in V \setminus \{v_s, v_e\}} \pi_i y_i \tag{3.1}$$

$$\text{s.a.:} \quad \sum_{m=0}^{|T_{ij}^m|} x_{ij}^m = x_{ij} \qquad (i,j) \in A \tag{3.2}$$

$$\sum_{(i,j) \in N^-(j)} x_{ij} = y_j \qquad j \in V \setminus \{v_s\} \tag{3.3}$$

$$\sum_{(v_s,j) \in N^+(v_s)} x_{v_s j} = \sum_{i \in N^-(v_e)} x_{i v_e} = 1 \tag{3.4}$$

$$\sum_{(i,k) \in N^-(k)} x_{ik} - \sum_{(k,j) \in N^+(k)} x_{kj} = 0 \qquad k \in V \setminus \{v_s, v_e\} \tag{3.5}$$

$$\sum_{(i,j) \in N^-(j)} \sum_{m=0}^{|T_{ij}^m|} (1 + \theta_{ij}^m) t_{ij}^m + \eta_{ij}^m x_{ij}^m = t_j \qquad j \in V \setminus \{v_s\} \tag{3.6}$$

$$\sum_{(i,j) \in N^+(i)} \sum_{m=0}^{|T_{ij}^m|} t_{ij}^m = t_i \qquad i \in V \setminus \{v_e\} \tag{3.7}$$

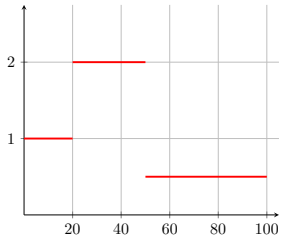
$$w_{ij}^m x_{ij}^m \le t_{ij}^m \le w_{ij}^{m+1} x_{ij}^m \qquad (i,j) \in A, m \in T_{ij}^m \tag{3.8}$$

$$\sum_{i \in V \setminus \{v_s, v_e\}} q_i y_i \le Q \tag{3.9}$$

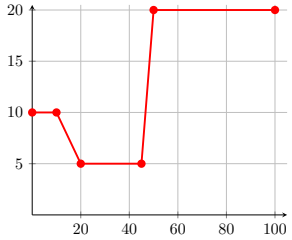
$$t_{ij}^m, t_i \ge 0 \qquad (i,j) \in A, m \in T_{ij}^m \tag{3.10}$$

$$y_i, x_{ij}, x_{ij}^m \in \{0,1\} \qquad (i,j) \in A, m \in T_{ij}^m \tag{3.11}$$

La función objetivo (3.1) minimiza la duración mientras maximiza los beneficios recolectados. Las restricciones (3.2) - (3.5) definen las variables x_{ij} y y_i en términos de x_{ij}^m y establecen que el vehículo debe partir desde v_s, terminar en v_e, e imponen las clásicas restricciones de conservación de flujo. Las restricciones (3.6) determinan la función de tiempo de finalización en el vértice j sumando el tiempo de partida desde i (es decir, t_{ij}^m) al tiempo de viaje desde i hasta j. El tiempo de viaje se modela como una función lineal a trozos del tiempo de partida, específicamente $\theta_{ij}^m \cdot t + \eta_{ij}^m$ cuando se parte en el tiempo t dentro del intervalo $[T_{ij}^m, T_{ij}^{m+1})$. Aquí, el término pendiente θ_{ij}^m captura el aumento dependiente del tiempo en el tiempo de viaje, mientras que la constante η_{ij}^m representa el tiempo base de viaje. Así, la pendiente $(1 + \theta_{ij}^m)$ en las restricciones considera el hecho de que partidas más tardías causan incrementos proporcionales en los tiempos de llegada, combinando tanto el efecto directo de la espera como el retraso del tiempo de viaje. Cabe destacar que esta restricción es una versión fortalecida de las consideradas en Sun et al. [21] y Montero et al. [17]. Las restricciones (3.7) expresan las variables t_i en términos de t_{ij}^m, y las restricciones (3.8) imponen la relación entre las variables t_{ij}^m y x_{ij}^m. Finalmente, las restricciones (3.10) y (3.11) establecen el dominio para todas las variables.



(a) Función de velocidad de viaje para (i, j).



(b) Función de tiempo de viaje $\tau_{ij}(t)$.

Fig. 3.1: Modelo de tiempo de viaje dependiente del tiempo ($d_{ij} = 10$).

3.3.2. Modelo 2: CTTBF

A continuación presentamos la nueva formulación Compact TTBF (CTTBF), que es una variación del modelo TTBF. La idea clave detrás del CTTBF es fortalecer la relajación y potencialmente reducir el número de variables y restricciones de la formulación TTBF considerada en Sun et al. [21] y Montero et al. [17] aprovechando alguna característica particular de las funciones de tiempo de viaje.

Dependiendo de la definición del perfil de velocidad y de la distancia del arco, d_{ij}, observamos que algunos de los períodos de tiempo de viaje corresponden a viajes que comienzan y terminan dentro del mismo intervalo de velocidad de viaje T_h, lo que significa que no hay cruce de frontera entre intervalos consecutivos de velocidad de viaje. Como resultado, en estos casos la función de tiempo de viaje para algunos de los intervalos de tiempo de viaje $T_m^{ij} \in T^{ij}$ se vuelve constante, y por lo tanto tiene pendiente $\theta_{ij}^m = 0$. Este es el caso para los primeros, terceros y quintos intervalos de tiempo en la Figura 3.1b.

Para cada $(i, j) \in A$, sea $\hat{T}^{ij} = \{m \in T^{ij} : \theta_{ij}^m = 0\}$ el subconjunto de intervalos donde la función de tiempo de viaje es constante. Además, definimos un nuevo conjunto de variables $\hat{t}_{ij} = t_i$ si el arco (i, j) es recorrido saliendo de i dentro de uno de los intervalos de tiempo $m \in \hat{T}^{ij}$, y $\hat{t}_{ij} = 0$ en caso contrario. Formalmente, podemos reescribir $\hat{t}_{ij} = \sum_{m \in \hat{T}^{ij}} t_{ij}^m$ y entonces las restricciones (3.6) y (3.7) pueden expresarse como

$$\sum_{(i,j)\in N^-(j)} \left(\hat{t}_{ij} + \sum_{m \in T^{ij} \setminus \hat{T}^{ij}} (1 + \theta_{ij}^m) t_{ij}^m + \sum_{m \in T^{ij}} \eta_{ij}^m x_{ij}^m \right) = t_j, \ j \in V \setminus \{v_s\} \tag{3.12}$$

$$\sum_{(i,j)\in N^+(i)} \left(\hat{t}_{ij} + \sum_{m \in T^{ij} \setminus \hat{T}^{ij}} t_{ij}^m \right) = t_i, \ i \in V \setminus \{v_e\} \tag{3.13}$$

Finalmente, las restricciones (3.8) también pueden adaptarse para incorporar las nuevas definiciones. Primero, observamos que para $m \in T^{ij} \setminus \hat{T}^{ij}$ las restricciones (3.8) deben mantenerse como parte de la formulación. Sin embargo, para $m \in \hat{T}^{ij}$, podemos reemplazarlas por las siguientes restricciones agregadas

$$\sum_{m \in \hat{T}^{ij}} w_{ij}^m x_{ij}^m \le \hat{t}_{ij} \le \sum_{m \in \hat{T}^{ij}} w_{ij}^{m+1} x_{ij}^m, \quad (i, j) \in A. \tag{3.14}$$

Recuerde que como máximo una variable x_{ij}^m, $m \in \hat{T}^{ij}$, puede tomar el valor 1 y por lo tanto $\hat{t}_{ij}$ captura el tiempo de partida correcto.

Estas definiciones permiten reducir el tamaño de la formulación resultante al reemplazar todas las variables t_{ij}^m para $m \in \hat{T}^{ij}$ por una sola variable $\hat{t}_{ij}$. Se obtiene una reducción similar en términos del número de restricciones al considerar las restricciones (3.14). Debe señalarse que el impacto de la reducción depende de la granularidad de los perfiles de velocidad y las distancias a recorrer. Este es el caso en las instancias de referencia disponibles en la literatura relacionada, que están definidas con el objetivo de capturar la congestión estándar de la hora pico en grandes ciudades.

Además, enfatizamos que esto también aparece como un escenario razonable para la planificación táctica, lo que motiva y justifica nuestro enfoque. La CTTBF se muestra a continuación

$$\text{mín} \quad t_{v_e} - t_{v_s} - \sum_{i \in V} \pi_i y_i \tag{3.15}$$

$$\text{s.a.:} \quad \sum_{m \in T^{ij}} x_{ij}^m = x_{ij} \qquad (i,j) \in A \tag{3.16}$$

$$\sum_{(i,j) \in N^-(j)} x_{ij} = y_j \qquad j \in V \setminus \{v_s\} \tag{3.17}$$

$$\sum_{v_s j \in N^+(v_s)} x_{v_s j} = y_{v_s} = y_{v_e} = 1 \tag{3.18}$$

$$\sum_{(i,k) \in N^-(k)} x_{ik} - \sum_{(k,j) \in N^+(k)} x_{kj} = 0 \qquad k \in V \setminus \{v_s, v_e\} \tag{3.19}$$

$$\sum_{(i,j) \in N^-(j)} \left(\hat{t}_{ij} + \sum_{m \in T^{ij} \setminus \hat{T}^{ij}} (1 + \theta_{ij}^m) t_{ij}^m + \sum_{m \in T^{ij}} \eta_{ij}^m x_{ij}^m \right) \leq t_j \qquad j \in V \setminus \{v_s\} \tag{3.20}$$

$$\sum_{(i,j) \in N^+(i)} \left(\hat{t}_{ij} + \sum_{m \in T^{ij} \setminus \hat{T}^{ij}} t_{ij}^m \right) = t_i \qquad i \in V \setminus \{v_e\} \tag{3.21}$$

$$w_{ij}^m x_{ij}^m \leq t_{ij}^m \leq w_{ij}^{m+1} x_{ij}^m \qquad (i,j) \in A, m \in T^{ij} \setminus \hat{T}^{ij} \tag{3.22}$$

$$\sum_{m \in \hat{T}^{ij}} w_{ij}^m x_{ij}^m \leq \hat{t}_{ij} \leq \sum_{m \in \hat{T}^{ij}} w_{ij}^{m+1} x_{ij}^m, \qquad (i,j) \in A \tag{3.23}$$

$$\hat{t}_{ij}, t_{ij}^m, t_i \geq 0 \qquad (i,j) \in A, m \in T^{ij} \tag{3.24}$$

$$y_i, x_{ij}, x_{ij}^m \in \{0,1\} \qquad (i,j) \in A, m \in T^{ij} \tag{3.25}$$

La función objetivo (3.15) minimiza la duración mientras maximiza los beneficios recolectados. Las restricciones (3.16) - (3.19) definen las variables x_{ij} y y_i en términos de x_{ij}^m y establecen que el vehículo debe salir del v_s, terminar en el v_e, e imponen las clásicas restricciones de conservación de flujo. Las restricciones (3.20) calculan la función de tiempo de finalización para el vértice j, si es visitado. Las restricciones (3.21) expresan las variables t_i en términos de t_{ij}^m y $\hat{t}_{ij}$, y las restricciones (3.22) y (3.23) imponen la relación entre las variables t_{ij}^m, $\hat{t}_{ij}$ y x_{ij}^m. Finalmente, las restricciones (3.24) y (3.25) establecen el dominio para todas las variables.

3.3.3. Equivalencia entre TTBF y CTTBF

Basándonos en algunos resultados experimentales preliminares, la versión reforzada de las relajaciones TTBF y CTTBF parecen ser equivalentes. En esta sección, mostramos que el modelo TTBF (3.1)-(3.11) es equivalente a CTTBF (3.15)-(3.25). Además, definimos un procedimiento para transformar soluciones de TTBF a soluciones de CTTBF y viceversa. Sea z^*_{TTBF} y z^*_{CTTBF} el valor óptimo de la función objetivo de la relajación LP para TTBF y CTTBF, respectivamente.

Proposición 3.1. $z^*_{TTBF} = z^*_{CTTBF}$.

Demostración. Recordemos que la CTTBF se construye agregando un subconjunto de las variables t_{ij}^m en un conjunto más pequeño de variables $\hat{t}_{ij}$. A pesar de esta compresión, la estructura general de la CTTBF permanece similar a la del TTBF original. En particular, las variables x_{ij}^m y y_i permanecen iguales. Por lo tanto, consideramos únicamente aquellas restricciones que involucran las variables t_{ij}^m. Las restricciones (3.20) de CTTBF reemplazan a las restricciones (3.6) de la formulación TTBF. Además, las restricciones (3.21) son la adaptación de las restricciones TTBF (3.7). Luego, las restricciones (3.22) permanecen iguales, sin embargo las restricciones (3.23) son una agregación de las restricciones originales (3.8) sobre $m \in \hat{T}^{ij}$.

Ahora, mostramos cómo transformar soluciones factibles entre TTBF y CTTBF manteniendo el mismo valor objetivo. La transformación de soluciones de TTBF a CTTBF se realiza utilizando la definición de $\hat{t}$. Dada una solución (fraccionaria) de la relajación lineal de TTBF $z = \langle x, y, t \rangle$, construimos una solución $z' = \langle x', y', t', \hat{t}' \rangle$ de la relajación lineal de CTTBF de la siguiente manera:

- $x' = x, y' = y$.
- $t'_i = t_i$ para cada $i \in V$.
- $t'^m_{ij} = t^m_{ij}$ para cada $(i,j) \in A, m \in T \setminus \hat{T}^{ij}$.
- $\hat{t}'_{ij} = \sum_{m \in \hat{T}^{ij}} t^m_{ij}$ para cada $(i,j) \in A$.

Reemplazando $\hat{t}'_{ij}$ por su definición en las ecuaciones (3.20) y (3.21) obtenemos las restricciones originales (3.6) y (3.7). Es fácil ver que las restricciones (3.23) también se cumplen para z'. Por lo tanto, dado que la función objetivo solo depende de las variables t_i y y_i y que tienen el mismo valor en z y en z', respectivamente, tenemos una solución válida z' para CTTBF con la misma función objetivo que z para TTBF.

A continuación nos concentramos en el caso restante. Sea $z' = \langle x', y', t', \hat{t}' \rangle$ una solución factible de CTTBF, construimos una solución $z = \langle x, y, t \rangle$, tal que sea factible y tenga el mismo valor objetivo, como sigue:

- $x = x', y = y'$.
- $t_i = t'_i$ para cada $i \in V$.
- $t^m_{ij} = t'^m_{ij}$ para cada $(i,j) \in A, m \notin \hat{T}^{ij}$.
- $t^m_{ij} = w^m_{ij} x^m_{ij} + \beta_{ij} \alpha^m_{ij}$ para cada $(i,j) \in A, m \in \hat{T}^{ij}$ donde
  - (i) $\beta_{ij} = \hat{t}'_{ij} - \sum_{m' \in \hat{T}^{ij}} w^{m'}_{ij} x^{m'}_{ij}$,
  - (ii) $\alpha^m_{ij} = \frac{w^{m+1}_{ij} x^m_{ij} - w^m_{ij} x^m_{ij}}{\sum_{m' \in \hat{T}^{ij}} w^{m'+1}_{ij} x^{m'}_{ij} - w^{m'}_{ij} x^{m'}_{ij}}$.

Primero, observar que $\beta_{ij} \alpha^m_{ij}$ es no negativo para cada arco $(i,j) \in A$ y $m \in \hat{T}^{ij}$ ya que las restricciones (3.23) imponen que $\sum_{m' \in \hat{T}^{ij}} w^{m'}_{ij} x^{m'}_{ij} \leq \hat{t}'_{ij}$. Por lo tanto, la primera desigualdad de las restricciones (3.8) se cumple para cada arco $(i,j) \in A$ y $m \in \hat{T}^{ij}$. Además, notar que las restricciones (3.23) también imponen que $\hat{t}'_{ij} \leq \sum_{m' \in \hat{T}^{ij}} w^{m'+1}_{ij} x^{m'}_{ij}$, y por lo tanto es fácil obtener, con alguna manipulación algebraica, que la segunda desigualdad de las restricciones (3.8) también se cumple para cada arco $(i,j) \in A$ y $m \in \hat{T}^{ij}$. Asimismo, se puede demostrar fácilmente que

$$\hat{t}'_{ij} = \sum_{m \in \hat{T}^{ij}} t^m_{ij}$$

Por lo tanto, dado que $\sum_{m \in \hat{T}^{ij}} t^m_{ij} = \hat{t}'_{ij}$, sabemos que las restricciones (3.6) y (3.7) son válidas porque (3.20) y (3.21) son válidas para z'. Finalmente, observar que las variables t_i, y_i son las mismas que t'_i, y'_i, lo que implica que el valor objetivo de ambas soluciones es el mismo. □

3.3.4. Adaptaciones de recursos y objetivos

El modelo puede adaptarse fácilmente para manejar diferentes funciones objetivo y recursos. Minimizar el makespan en lugar de la duración se puede lograr fácilmente estableciendo $t_{v_s} = 0$. Alternativamente, se pueden recuperar variantes de TSP donde cada vértice debe ser visitado exactamente una vez estableciendo $y_i = 1$ para $i \in V$. Además, las ventanas de tiempo pueden modelarse con una restricción que impone una limitación sobre las variables t_i.

$$a_i y_i \leq t_i \leq b_i y_i \quad i \in V \tag{3.26}$$

Para modelar las demandas y la capacidad del vehículo, por otro lado, tenemos dos enfoques diferentes dependiendo de si las demandas pueden tomar valores negativos o no. Si todas las

demandas son positivas, entonces basta con requerir que la suma de las demandas de las rutas no exceda la capacidad del vehículo.

$$\sum_{i \in V} y_i d_i \leq Q \tag{3.27}$$

Con respecto a las variantes de orienteering, debemos incorporar una restricción que limite la duración de la ruta. Nuestra restricción es ligeramente diferente de la considerada en Verbeeck et al. [25] ya que modelamos explícitamente el tiempo de partida desde un vértice $i \in V$ con la variable t_i. Así, la duración de la ruta puede limitarse con la siguiente restricción

$$t_{v_e} - t_{v_s} \leq t_{\max}. \tag{3.28}$$

Nótese que imponer un límite para el makespan de la ruta se sigue directamente dado que $t_{v_s} = 0$.

Para el Problema de Tiempo de Viaje Variable con Recolecciones y Entregas (TDPTP, por sus siglas en inglés), tener en cuenta que Sun et al. [21] consideran la variante uno a uno y que la demanda para la entrega se modela como una cantidad negativa. Nosotros manejamos el caso de demandas negativas siguiendo el mismo enfoque. Sea $Q_i \geq 0$ la variable que representa la demanda transportada por el vehículo después de atender al cliente i. Debemos verificar que cualquier Q_i sea menor o igual que la capacidad del vehículo. Recordar que una solicitud r_k está definida por el vértice de recolección k y el vértice de entrega $n+k$, $r_k = (k, n+k)$, y la correspondiente restricción de precedencia debe cumplirse si se visitan los vértices k y $n+k$.

$$y_k = y_{n+k} \quad k \in R \tag{3.29}$$

$$t_k \leq t_{n+k} \quad k \in R \tag{3.30}$$

$$Q_i + q_j \leq Q_j + M(1 - x_{ij}) \quad (i,j) \in A \tag{3.31}$$

$$0 \leq Q_i \leq \min\{Q, Q + q_i\} \quad i \in V \tag{3.32}$$

3.4. Desigualdades válidas

En esta sección definimos distintas familias de desigualdades válidas para CTTBF con el fin de mejorar la calidad de la cota inferior proporcionada por la relajación LP, explotando explícitamente la estructura dependiente del tiempo. Dado que sólo consideran las variables x_{ij}^m, estas desigualdades también son válidas para TTBF. En las secciones 3.4.1 y 3.4.2 presentamos familias de desigualdades válidas de la literatura que también son válidas para CTTBF. Además, las secciones 3.4.3, 3.4.4 y 3.4.5 presentan nuevos resultados teóricos sobre desigualdades válidas que explotan la dependencia temporal. En la Sección 3.4.3 introducimos dos familias de desigualdades válidas que generalizan desigualdades tradicionales para el caso dependiente del tiempo. Luego, en la Sección 3.4.4 discutimos la relación entre algunas de ellas así como con algunas familias conocidas de desigualdades válidas. Finalmente, en la Sección 3.4.5 presentamos algoritmos en tiempo polinomial para el problema de separación de las nuevas desigualdades válidas.

3.4.1. Desigualdades Generalized Cut Set

La primera familia de desigualdades que presentamos se conoce como *Generalized Cut Set* (GCS, por sus siglas en inglés), explicada en Taccari [23], que es una generalización de las clásicas *Subtour Elimination Constraints* (SEC, por sus siglas en inglés) para el caso en que no es necesario visitar a todos los clientes. Dado un conjunto $S \subseteq V \setminus \{v_s, v_e\}$ con $|S| \geq 2$, y un vértice $k \in S$, la desigualdad GCS se define por

$$\sum_{(i,j) \in N^+(S)} x_{ij} \geq y_k \tag{3.33}$$

En términos generales, estas desigualdades imponen que, para cualquier subconjunto S de vértices que no incluya al depósito, donde al menos uno de ellos es visitado, debe existir al menos un arco que salga de S; de lo contrario, se generaría un subtour. Es fácil ver que esta familia de desigualdades es válida para el CTTBF.

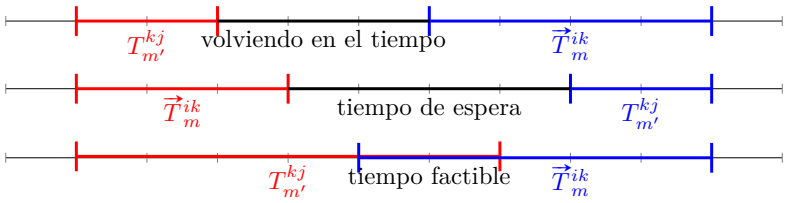


Fig. 3.2: Ejemplo sobre el concepto de sucesor de arco dependiente del tiempo.

3.4.2. Desigualdades del TSP con restricciones de precedencia

Una de las variantes consideradas para evaluar el desempeño del algoritmo BC es el TDTSPTW. Bajo la presencia de ventanas de tiempo y con la restricción de que todos los clientes deben ser visitados, se pueden inferir precedencias entre los clientes utilizando esta información. Se han propuesto varias familias de desigualdades válidas que explotan estas precedencias, las cuales son bastante efectivas para mejorar la relajación LP incluso cuando se consideran tiempos de viaje dependientes del tiempo, como se muestra en Montero et al. [17]. Así, siguiendo su enfoque, también incluimos las conocidas desigualdades π, σ y (π,σ) al abordar el TDTSPTW.

3.4.3. Desigualdades dependientes del tiempo

Las siguientes familias de desigualdades válidas explotan la estructura particular de la función de tiempo de viaje dependiente del tiempo. Para mayor simplicidad, primero introducimos alguna notación utilizada a lo largo de la sección. Recuerde que $\hat{A}$ es el conjunto de todos los arcos dependientes del tiempo, y T^{ij}_m denota el intervalo en el que ocurre la partida desde el cliente i para el período de tiempo m. De manera similar, definimos el intervalo $\vec{T}^{ij}_m = [w^m_{ij} + \tau_{ij}(w^m_{ij}), w^{m+1}_{ij} + \tau_{ij}(w^{m+1}_{ij})]$ cuando es posible llegar al cliente j si se parte del cliente i en T^{ij}_m.

Supongamos que P es una solución factible, y $t \in [0,T]$ el mejor tiempo de partida para la ruta P, que como se señaló en la Sección 3.2 no contiene tiempos de espera. Supongamos que los arcos $(i,j),(j,k) \in A$ se utilizan en P partiendo de cada uno en los períodos de tiempo $m \in T^{ij}$, $m' \in T^{jk}$ respectivamente. Intuitivamente, dado que no se requieren tiempos de espera, entonces m y m' deben estar de alguna manera relacionados. En particular, el tiempo en que es factible llegar a j si se parte de i en el período de tiempo m debería coincidir con el tiempo factible para partir de j hacia k en el período de tiempo m'. Usamos esta idea para definir la noción de *sucesor de arco.*

Considerar un par de arcos con tiempos variables (i,j,m), $(j,k,m') \in \hat{A}$, decimos que (j,k,m') es un *sucesor* de (i,j,m) si $\vec{T}^{ij}_m \cap T^{jk}_{m'} \neq \varnothing$. Además, definimos el conjunto de sucesores $\hat{N}^+(i,j,m)$ del arco $(i,j,m) \in \hat{A}$ como

$$\hat{N}^+(i,j,m) = \{(j,k,m') \in \hat{A} : \vec{T}^{ij}_m \cap T^{jk}_{m'} \neq \varnothing\} \tag{3.34}$$

De manera más general, extendemos la definición de sucesores de un conjunto de arcos $F \subseteq \hat{A}$ como

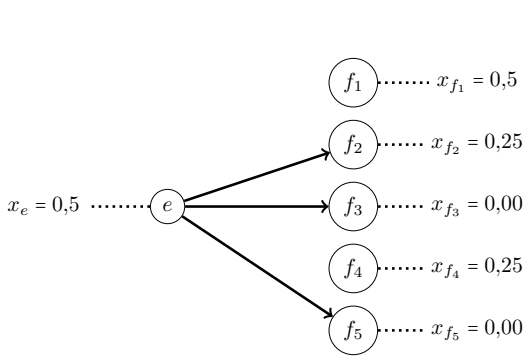
$$\hat{N}^+(F) = \bigcup_{(i,j,m)\in F} \hat{N}^+(i,j,m) \setminus F, \tag{3.35}$$

lo cual considera los arcos que son sucesores de al menos un arco en F pero que no pertenecen a F. De manera similar, podemos definir los predecesores de un arco como

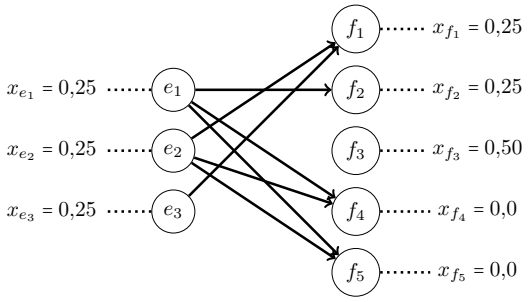
$$\hat{N}^-(j,k,m') = \{(i,j,m) \in \hat{A} : \vec{T}^{ij}_m \cap T^{jk}_{m'} \neq \varnothing\}, \tag{3.36}$$

y su versión generalizada

$$\hat{N}^-(F) = \bigcup_{(j,k,m')\in F} \hat{N}^-(j,k,m') \setminus F, \tag{3.37}$$



(a) Desigualdad single flow con dependencia temporal violada para $e = (i, j, m) \in \hat{A}$, donde $f_1 \dots f_5 \subseteq \hat{A}$ son todos los arcos con dependencia temporal que parten de j.



(b) Desigualdad de flujo con dependencia temporal violada para los arcos F que entran al vértice $k \in V$ definida como $F = \{e_1 = (i_1, j_1, m_1), \cdots, e_3 = (i_3, j_3, m_3)\} \subseteq \hat{A}$, donde $f_1 \dots f_5 \subseteq \hat{A}$ son todos los arcos con dependencia temporal que salen de i_1, i_2 o i_3.

Fig. 3.3: Ejemplo de las desigualdades de flujo con dependencia temporal.

La Figura 3.2 muestra tres situaciones diferentes. En las dos primeras, ambos arcos no pueden aparecer en la misma solución ya que implican tiempos de espera o requieren viajar hacia atrás en el tiempo. El último escenario representa un ejemplo de dos arcos que pueden estar simultáneamente activos en una solución óptima al llegar a k en algún punto dentro del intervalo de tiempo factible.

Desigualdades de flujo dependientes del tiempo

En esta sección presentamos una nueva familia de desigualdades válidas que se basa en las bien conocidas restricciones de conservación de flujo. La idea es que las restricciones de conservación de flujo deben cumplirse en cada vértice visitado, pero también el flujo debe ser consistente en términos de dependencia temporal.

La intuición detrás de las desigualdades de flujo con dependencia temporal es la siguiente. Considere un dígrafo $\hat{L}(D) = \langle V^L, A^L \rangle$ donde hay un vértice en V^L para cada arco con dependencia temporal en $\hat{A}$ y hay un arco desde el vértice $e \in V^L$ al vértice $f \in V^L$ si $f \in \hat{N}^+(e)$. Llamamos a $\hat{L}(D)$ el dígrafo línea con dependencia temporal de G y, para evitar confusiones, notaremos sus arcos como $e \to f$. Claramente, todas las soluciones enteras factibles sin tiempos de espera deben ser caminos en $\hat{L}(D)$, de lo contrario incluirían una secuencia de arcos con dependencia temporal que obligarían a tiempos de espera. Como resultado, el flujo debe conservarse en $\hat{L}(D)$. No distinguimos los vértices de $\hat{L}(D)$ de los arcos con dependencia temporal $\hat{A}$ ya que existe una biyección entre ellos. La Figura 3.3 muestra un ejemplo de una solución fraccionaria válida en términos de las restricciones clásicas de flujo (3.19) que se convierte en una solución inválida al considerar la dependencia temporal. Los arcos en negrita representan los arcos $e \to f$ definidos en $\hat{L}(D)$. A continuación mostramos en la Proposición 3.2 que el flujo debe conservarse entre un arco con dependencia temporal y sus sucesores.

Proposición 3.2. *Las desigualdades single flow con dependencia temporal definidas por*

$$x_{ik}^m \leq \sum_{(k,j,m') \in \hat{N}^+(i,k,m)} x_{kj}^{m'} \quad (i,k,m) \in \hat{A}, k \neq v_e \tag{3.38}$$

son válidas para CTTBF.

Demostración. Sea $\bar{x}$ una solución entera que representa el camino P, y sea $(i,k) \in A$ un arco. El caso de $\bar{x}_{ik}^m = 0$ es trivial. Por el contrario, $\bar{x}_{ik}^m = 1$ significa que el arco (i,k,m) pertenece a P. Dado que P es finito y elemental, entonces existe un arco $(k,j,m') \in P$ que sigue a (i,k,m). Se deduce que el arco (k,j,m') debe ser un sucesor de (i,k,m), de lo contrario no estaría en una solución óptima. □

Extendemos esta noción tomando un grupo de arcos que entran a algún vértice $k \in V \setminus \{v_e\}$ en lugar de uno solo. Sea $F \subseteq \hat{A}$ un conjunto de arcos que entran a k, es decir, $(i,j,m) \in F$ si y sólo si $j = k$. Sabemos que el flujo que entra y sale del arco k debe ser igual, por lo tanto, el flujo que entra a k a través de los arcos en F debe salir por sus sucesores $\hat{N}^+(F)$. Nótese que no consideramos $k \in \{v_s, v_e\}$ ya que el flujo no se conserva en esos vértices.

Proposición 3.3 (TDFI)**.** *Las* Desigualdades de Flujo con Tiempos Dependientes (TDFI, por sus siglas en inglés)

$$\sum_{(i,k,m)\in F} x_{ik}^m \leq \sum_{(k,j,m')\in \hat{N}^+(F)} x_{kj}^{m'} \quad k \in V \setminus \{v_s, v_e\}, F \subseteq \{(i,k,m) \in \hat{A}\} \tag{3.39}$$

son válidas para CTTBF.

Demostración. La demostración es análoga a la demostración de la Proposición 3.2 si notamos que a lo sumo una variable puede ser seleccionada en cada lado de la desigualdad en una solución entera debido a las restricciones (3.16) y (3.17). □

A continuación, ilustramos algunas situaciones que son capturadas por estas desigualdades. Por ejemplo, supongamos que tenemos un cliente $k \in V \setminus \{v_s, v_e\}$ y un subconjunto $\hat{A}_{in} \in \hat{A}$ de los arcos que llegan a k antes del tiempo $t \in [0,T]$, y un subconjunto $\hat{A}_{out} \subseteq \hat{A}$ de los arcos que parten del vértice k estrictamente después del tiempo t. Es fácil ver que a lo sumo un arco en $\hat{A}_{in} \cup \hat{A}_{out}$ puede ser seleccionado en una solución entera óptima, porque de lo contrario la solución incurriría en tiempos de espera. Este caso particular está capturado por las TDFI al elegir $F = \{(i,k,m) \in \hat{A} : \text{máx}(\vec{T}_m^{ik}) \leq t\}$. De manera similar, un conjunto de arcos que llegan a k después del tiempo t y otro con arcos que parten de k antes del tiempo t no pueden contener dos arcos seleccionados al mismo tiempo. Esta situación representa soluciones que viajan hacia atrás en el tiempo. Nuevamente, podemos manejar tales escenarios eligiendo conjuntos apropiados de arcos F para definir las TDFI.

El impacto de la TDFI en términos de la relajación LP puede caracterizarse teóricamente extendiendo la formulación MILP de la Sección 3.3 con variables y restricciones adicionales. Sea $e = (i,j,m) \in \hat{A}$ un arco con dependencia temporal. Por simplicidad de notación, en esta sección denotamos las variables x_{ij}^m como x_e. Introducimos las *variables de flujo con dependencia temporal* $\alpha_{ef} \in \{0,1\}$ para $e, f \in \hat{A}$ y $f \in \hat{N}^+(e)$ que toman valor 1 si el arco f es inmediatamente después del arco e en la solución, y 0 en caso contrario.

La CTTBF puede fortalecerse añadiendo las siguientes restricciones para limitar el flujo de las variables con las siguientes restricciones. Denominamos a esta formulación extendida CTTBF-EXT.

$$x_e = \sum_{f\in\hat{N}^+(e)} \alpha_{ef} \quad \forall e \in \{(i,j,m) \in \hat{A} \mid j \neq v_e\} \tag{3.40}$$

$$x_e = \sum_{f\in\hat{N}^-(e)} \alpha_{fe} \quad \forall e \in \{(i,j,m) \in \hat{A} \mid i \neq v_s\} \tag{3.41}$$

$$\alpha_{ef} \in \{0,1\} \quad \forall e \in \hat{A}, f \in \hat{N}^+(e). \tag{3.42}$$

Las restricciones (3.40) y (3.41) establecen la equivalencia entre las variables de arco con dependencia temporal y las variables de flujo con dependencia temporal imponiendo la conservación del flujo en los arcos con dependencia temporal. Finalmente, las restricciones (3.42) indican que las variables de flujo con dependencia temporal deben ser binarias.

Observamos que el número de variables de flujo con dependencia temporal α_{ef} incrementa significativamente el tamaño de la formulación, lo cual puede volverse muy restrictivo desde el punto de vista computacional debido a la gran cantidad de variables. A continuación, mostramos que la relajación LP de CTTBF-EXT es equivalente al CTTBF fortalecido con las TDFI, al que denominamos CTTBF-TDFI. Primero, demostramos el siguiente lema.

Lema 3.1. *Sea $k \in V \setminus \{v_s, v_e\}$ un cliente, $F_k^{in} = \{(i,k,m) \in \hat{A}\}$ y $F_k^{out} = \{(k,j,m) \in \hat{A}\}$ los conjuntos de arcos con dependencia temporal que entran y salen de k, respectivamente, y sea*

$\langle \bar{x}_{ij}, \bar{x}^m_{ij}, \bar{y}_i, \bar{t}_i, \bar{t}^m_{ij} \rangle$ una solución factible de la relajación LP de CTTBF-TDFI. Entonces, existe una asignación para las variables α_{ef}, $e \in F^{in}_k, f \in \hat{N}^+(e)$, tal que las restricciones (3.40) se cumplen para todo $e \in F^{in}_k$ y las restricciones (3.41) se cumplen para todo $f \in F^{out}_k$.

Demostración. La clave de la demostración es encontrar el flujo máximo de una red específica para hallar valores adecuados para α. Sea $N = \langle N_V = \{s, t, N_{in}, N_{out}\}, N_A, c\rangle$ una red con vértice fuente s, vértice objetivo t, vértices

$$N_{in} = \{u_e \mid e \in F^{in}_k\}, N_{out} = \{w_f \mid f \in F^{out}_k\},$$

y arcos

$$N_A = \{(s, u_e) \mid u_e \in N_{in}\} \cup \{(u_e, w_f) \mid f \in \hat{N}^+(e)\} \cup \{(w_f, t) \mid w_f \in N_{out}\}.$$

La capacidad se establece como $c(s, u_e) = \bar{x}_e$ para todo $(s, u_e) \in N_A$, $c(u_e, w_f) = +\infty$ para todo $(u_e, w_f) \in N_A$, y $c(w_f, t) = \bar{x}_f$ para todo $(w_f, t) \in N_A$.

Sea $\mathcal{F}$ el flujo máximo de la red N. Se puede verificar fácilmente que en $\mathcal{F}$ el flujo que sale de los vértices $u_e \in N_{in}$ es como máximo $\bar{x}_e$, y el flujo que entra a los vértices $w_f \in N_{out}$ es como máximo $\bar{x}_f$. Además, si el valor de $\mathcal{F}$ es $\bar{y}_k = \sum_{e \in F^{in}_k} \bar{x}_e = \sum_{f \in F^{out}_k} \bar{x}_f$, entonces las igualdades se cumplen. En ese caso, podemos asignar $\alpha_{ef} = \mathcal{F}(u_e, w_f)$ y sabemos que las restricciones (3.40), (3.41) se satisfacen.

El resto de la demostración consiste en mostrar que efectivamente el valor de $\mathcal{F} = \bar{y}_k$. Sea $\bar{S}, \bar{T}$ un corte mínimo, $c(\bar{S}, \bar{T})$ es igual al valor de $\mathcal{F}$. Claramente, $c(\bar{S}, \bar{T}) \leq \bar{y}_k$ ya que podemos tomar el corte $S = \{s\}$ y $T = N_V \setminus S$. Nos queda por mostrar que $c(\bar{S}, \bar{T}) \geq y_k$. Observar que los arcos (u_e, w_f) no pueden ser parte del corte mínimo $\bar{S}, \bar{T}$, ya que de lo contrario $c(\bar{S}, \bar{T}) = +\infty$. Entonces, podemos escribir la capacidad del corte mínimo como:

$$c(\bar{S}, \bar{T}) = \sum_{u_e \in \bar{T}} c(s, u_e) + \sum_{w_f \in \bar{S}} c(w_f, t). \tag{3.43}$$

Considere $F' = \{e \in F^{in}_k \mid u_e \in \bar{S}\}$. Observar que dado un vértice $e \in F'$, entonces para cada sucesor $f \in \hat{N}^+(e)$ el vértice w_f también pertenece a $\bar{S}$. Por lo tanto,

$$\sum_{w_f \in \bar{S}} c(w_f, t) \geq \sum_{f \in \hat{N}^+(F')} c(w_f, t) = \sum_{f \in \hat{N}^+(F')} \bar{x}_f \geq \sum_{e \in F'} \bar{x}_e, \tag{3.44}$$

donde la última desigualdad se cumple dado que la formulación incluye la TDFI asociada con F'. Finalmente, combinando (3.43) y (3.44) obtenemos

$$c(\bar{S}, \bar{T}) \geq \sum_{u_e \in \bar{T}} c(s, u_e) + \sum_{e \in F'} \bar{x}_e = \sum_{u_e \in \bar{T}} c(s, u_e) + \sum_{u_e \in \bar{S}} c(s, u_e) = \bar{y}_k. \tag{3.45}$$

□

Sea z^*_{EXT} y z^*_{TDFI} la función objetivo de la relajación LP de CTTBF-EXT y CTTBF-TDFI, respectivamente. Tenemos el siguiente resultado.

Proposición 3.4. $z^*_{EXT} = z^*_{TDFI}$.

Demostración. La demostración se divide en dos casos. Mostramos que podemos transformar una solución $\widetilde{z}$ de CTTBF-EXT en una solución $\bar{z}$ de CTTBF-TDFI que tenga el mismo valor objetivo y viceversa.

- **Caso 1**: Sea $\widetilde{z} = \langle \widetilde{x}_{ij}, \widetilde{x}^m_{ij}, \widetilde{y}_i, \widetilde{t}_i, \widetilde{t}^m_{ij}, \widetilde{\alpha}_{ef} \rangle$ una solución factible de CTTBF-EXT. Definimos $\bar{z} = \langle \bar{x}_{ij}, \bar{x}^m_{ij}, \bar{y}_i, \bar{t}_i, \bar{t}^m_{ij} \rangle$ simplemente descartando las variables de flujo con dependencia temporal $\widetilde{\alpha}_{ef}$. Es evidente que $\bar{z}$ satisface las restricciones (3.15) - (3.25). Para mostrar que es factible para CTTBF-TDFI, queda por demostrar que se cumplen todas las TDFI.

$$\sum_{e \in F} \bar{x}_e = \sum_{e \in F} \widetilde{x}_e = \sum_{e \in F} \sum_{f \in \hat{N}^+(e)} \widetilde{\alpha}_{ef} = \sum_{(e,f) \mid e \in F, f \in \hat{N}^+(e)} \widetilde{\alpha}_{ef} = \sum_{(e,f) \mid f \in \hat{N}^+(F), e \in \hat{N}^-(f)} \widetilde{\alpha}_{ef} = \tag{3.46}$$
$$\sum_{f \in \hat{N}^+(F)} \sum_{e \in \hat{N}^-(f) \cap F} \widetilde{\alpha}_{ef} \leq \sum_{f \in \hat{N}^+(F)} \sum_{e \in \hat{N}^-(f)} \widetilde{\alpha}_{ef} = \sum_{f \in \hat{N}^+(F)} \widetilde{x}_f = \sum_{f \in \hat{N}^+(F)} \bar{x}_f$$

Dado que las variables $\bar{t}_i = \widetilde{t}_i$ y $\bar{y}_i = \widetilde{y}_i$, el valor objetivo de ambas soluciones es igual.

- **Caso 2**: Sea $\bar{z} = \langle \bar{x}_{ij}, \bar{x}^m_{ij}, \bar{y}_i, \bar{t}_i, \bar{t}^m_{ij} \rangle$. Construimos la solución $\widetilde{z} = \langle \widetilde{x}_{ij}, \widetilde{x}^m_{ij}, \widetilde{y}_i, \widetilde{t}_i, \widetilde{t}^m_{ij}, \widetilde{\alpha}_{ef} \rangle$ con el mismo valor para todas las variables presentes en ambas formulaciones. A continuación mostramos que existe una asignación factible para las variables $\widetilde{\alpha}_{ef}$ tal que se cumplan las restricciones (3.40) y (3.41).
  Sea $k \in V \setminus \{v_s, v_e\}$ un cliente cualquiera, y definamos $F = \{(i,k,m) \in \hat{A}\}$, $F' = \hat{N}^+(F) = \{(k,j,m) \in \hat{A}\}$ como el conjunto de arcos con dependencia temporal que entran y salen de k, respectivamente. Debido al Lema 3.1, sabemos que existen valores para todas las $\widetilde{\alpha}_{ef}$ con $e \in F, f \in \hat{N}^+(e)$ tales que las restricciones (3.40) se cumplen para todo $e \in F$ y las restricciones (3.41) se cumplen para todo $f \in F'$. Dado que k es arbitrario e induce una partición del conjunto de variables $\widetilde{\alpha}_{ef}$, obtenemos una asignación factible para todas las variables.

□

Desigualdades Generalized Cut Set con dependencia temporal

La última familia de desigualdades es una generalización de las GCS para manejar la dependencia temporal. La idea detrás de esta familia de desigualdades es cortar soluciones fraccionarias que inducen un subtour en el dígrafo formado por los arcos con dependencia temporal. En la Proposición 3.5, introducimos la primera familia de desigualdades, que luego se fortalece en la Proposición 3.6.

Siguiendo la idea de las GCS decimos que, dado un conjunto de arcos S que no entran a v_e, el flujo de los sucesores de S debe ser al menos el flujo que pasa por cada arco en S. La siguiente familia de desigualdades válidas tiene como objetivo prohibir esta situación en soluciones factibles de la relajación LP.

Proposición 3.5. *Las desigualdades definidas como*

$$x^m_{ik} \leq \sum_{(j,l,m') \in \hat{N}^+(S)} x^{m'}_{jl} \quad S \subseteq \{(i,j,m) \in \hat{A} : j \neq v_e\}, (i,k,m) \in S \tag{3.47}$$

son válidas para el CTTBF.

Demostración. Como en el caso del GCS, la clave de esta demostración es notar que ningún arco que entra al depósito final v_e está seleccionado en S, y por lo tanto, si el arco $(i,k,m) \in S$ está seleccionado, entonces debe existir un camino de sucesores desde (i,k,m) hasta el depósito final que esté seleccionado. Sea $S \subseteq \{(i,j,m) \in \hat{A} : j \neq v_e\}$ un conjunto de arcos, $(i,k,m) \in S$ un arco incluido en S, y $\bar{x}$ una solución entera de CTTBF. Si $\bar{x}^m_{ik} = 0$, la desigualdad se satisface trivialmente. De lo contrario, sea $\bar{x}^m_{ik} = 1$, entonces, dado que $\bar{x}$ representa una solución entera factible, está asociada con un camino $e_1 \to \cdots \to e_r \to (i,k,m) \to e_{r+1} \to \cdots \to e_s$ de vértices en $\hat{L}(D)$ donde e_s llega al depósito final. Dado que $(i,k,m) \in S$ y $e_s \notin S$, debe existir al menos un vértice e_l entre e_{r+1} y e_s que no esté en S. Sea l el índice de dicho primer arco. Tenemos que $e_{l-1} \in S$, y $e_l \notin S$, y también que $e_l \in \hat{N}^+(e_{l-1})$, por lo tanto $e_l \in \hat{N}^+(S)$. Como resultado, el lado derecho de la desigualdad es al menos uno. □

Como se mencionó anteriormente, las desigualdades (3.47) pueden fortalecerse considerando que solo un arco que entra a cada vértice $k \in V$ puede ser seleccionado en cualquier solución entera. La Proposición 3.6 muestra esta nueva familia de desigualdades y demuestra su validez.

Proposición 3.6 (TDCSI)**.** *Las desigualdades Time-Dependent Cut Set (TDCSI) definidas como*

$$\sum_{(i,k,m) \in S \cup \hat{N}^+(S)} x^m_{ik} \leq \sum_{(i,j,m) \in \hat{N}^+(S)} x^m_{ij} \quad k \in V, S \subseteq \{(i,j,m) \in \hat{A} : j \neq v_e\} \tag{3.48}$$

son válidas para el CTTBF.

Demostración. Notar que solo un arco $(i,k,m) \in \hat{A}$ que entra al vértice $k \in V$ puede ser seleccionado en cualquier solución entera. Por lo tanto, sabemos que el valor del lado izquierdo es como máximo uno. El caso cuando esta expresión es exactamente 0 se satisface trivialmente, así que consideramos solo el caso cuando el valor del lado izquierdo es uno. Entonces, tenemos dos posibles escenarios

- si $\bar{x}^m_{ik} = 1$ para algún $(i,k,m) \in S$, entonces por la Proposición 3.5 el lado derecho es al menos uno.
- si $\bar{x}^m_{ik} = 0$ para todos los $(i,k,m) \in S$, entonces las variables del lado izquierdo son un subconjunto del lado derecho, por lo que la desigualdad es válida.

La versión fortalecida de la TDCSI, por lo tanto, es válida para CTTBF. □

3.4.4. Relación entre familias de desigualdades válidas

En esta sección, discutimos la relación entre diferentes familias de desigualdades. Primero, mostramos que las TDCSI generalizan tanto a las TDFI como a las GCS. Luego, establecemos una conexión entre las TDCSI y algunas desigualdades válidas propuestas para otra variante del TDTSP.

Desigualdades válidas para el CTTBF

Comenzamos mostrando que la TDFI puede expresarse como un caso particular de la TDCSI.

Observación 3.1. *Las desigualdades TDFI son un caso particular de la TDCSI.*

Demostración. Sea $k \in V \setminus \{v_s, v_e\}$ un cliente y $F \subseteq \{(i,k,m) \in \hat{A}\}$ un conjunto de arcos con dependencia temporal que entran a k y definen una TDFI. Se puede verificar fácilmente que la desigualdad anterior es equivalente a la TDCSI definida por $S = F$ y el vértice k. □

Observar que, aunque la TDCSI implica la TDFI, en la Sección 3.4.5 mostramos que la TDFI tiene un procedimiento de separación rápido, mientras que la separación de la TDCSI se realiza utilizando algoritmos de corte mínimo que usualmente tienen tiempos de ejecución más lentos. A continuación, también mostramos cómo las clásicas GCS pueden verse como un caso particular de la TDCSI. La idea detrás de este análisis es el hecho de que ambas desigualdades intentan separar soluciones con subtours.

Proposición 3.7. *Las TDCSI generalizan las GCS.*

Demostración. Sea $S \subseteq V \setminus \{v_s, v_e\}$ un conjunto que contiene al menos dos clientes, y sea $k \in S$ un cliente que define una desigualdad GCS. Sea $S' = \{(i,j,m) \in \hat{A} : j \in S\}$ como el conjunto de arcos con dependencia temporal que definen una TDCSI.

Primero, notar que las restricciones (3.16) y (3.17) implican

$$y_k = \sum_{(i,k,m) \in S' \cup \hat{N}^+(S')} x^m_{ik}, \tag{3.49}$$

ya que, por definición, todos los arcos que entran a $k \in S$ pertenecen a S'.

Además, tenemos

$$\hat{N}^+(S') = \{(i,j,m) \in \hat{A} : (i,j) \in N^+(S)\}. \tag{3.50}$$

Para ver esto, considere cualquier arco $(i,j) \in N^+(S)$. Por definición, $i \in S$, por lo que todos los arcos con dependencia temporal $(k,i,m) \in \hat{A}$ que entran al vértice i pertenecen a S'. Dado que $j \notin S$, y que cada arco $(i,j,m) \in \hat{A}$ debe ser sucesor de al menos un arco que entra a i (de lo contrario puede ser eliminado), se sigue que $(i,j,m) \in \hat{N}^+(S')$.

Finalmente, es fácil ver que cualquier arco $(i,j,m) \in \hat{N}^+(S')$ satisface $(i,j) \in N^+(S)$, porque i debe ser la cabeza de algún arco en S', y j no puede pertenecer a S. □

Conexiones con otros problemas

Discutimos brevemente la relación entre el TDCSI con otros problemas con dependencia temporal, en particular con el TDTSP, donde el costo de un arco depende de su posición en la ruta (véase, por ejemplo, Picard et al. [18]). El problema puede definirse en una red multipartita particular, donde los vértices de esta red codifican una combinación entre los vértices del problema original y su posición en la ruta. Por lo tanto, existe una definición implícita de *sucesores* que comparte algunas de las ideas propuestas en este capítulo.

Algunos enfoques exactos recientes para esta variante del TDTSP se centran en resultados poliedrales y algoritmos BC, y se pueden establecer algunas conexiones con estos problemas. Al considerar una definición apropiada del conjunto de sucesores de un arco con dependencia temporal, las llamadas *desigualdades de ciclo con dependencia temporal* propuestas en Miranda-Bront et al. [16] son generalizadas por las TDCSI. Otro ejemplo es las *restricciones de flujo admisibles* propuestas en Abeledo et al. [1] que también presentan una estructura similar, aunque su definición de alguna manera codifica implícitamente la estructura particular del grafo multipartito. Además, estas dos familias de desigualdades también comparten características con algunas desigualdades válidas para el Generalized TSP.

La relación entre estas familias de desigualdades válidas excede el alcance de este capítulo, y podrían seguirse investigaciones adicionales para establecer con precisión las conexiones entre los diferentes problemas. Sin embargo, destacamos que nuestra noción de sucesores captura la naturaleza de la dependencia temporal y depende directamente de los tiempos de viaje entre clientes.

3.4.5. Rutinas de separación

En esta sección presentamos las rutinas de separación para las desigualdades introducidas en las secciones anteriores. Primero, mencionamos brevemente las rutinas de separación consideradas para las desigualdades válidas adaptadas de problemas relacionados. Luego, presentamos los algoritmos de separación para las nuevas familias, que se explican en detalle en 3.A.1.

Desigualdades válidas de problemas relacionados

Las GCS se separan siguiendo el enfoque estándar descrito en Taccari [23]. En cuanto a las π, σ y (π,σ) seguimos el enfoque original propuesto en Balas et al. [4]. Cabe destacar que, en ambos casos, los algoritmos de separación se basan en resolver una serie de problemas de corte mínimo en un subgrafo.

Desigualdades de Flujo con Dependencia Temporal

La complejidad en el peor caso del problema de separación para las TDFI es polinomial. El algoritmo utiliza un enfoque tradicional de programación dinámica, similar al estándar para el problema del Subset Sum. Recuerde que nuestro objetivo es maximizar la violación de la TDFI definida en (3.39), lo que se traduce formalmente en $\max_{F,k} \sum_{(i,k,m)\in F} x^m_{ik} - \sum_{(k,j,m')\in \hat{N}^+(F)} x^{m'}_{kj}$ con F, k cumpliendo las condiciones de la desigualdad.

Sea $k \in V \setminus \{v_e\}$ un vértice, $I = \{e_1, \cdots, e_r\}, O = \{f_1, \cdots, f_s\}$ los conjuntos de arcos con dependencia temporal que entran y salen de k respectivamente, y $\bar{x}$ una solución fraccionaria de CTTBF. Supongamos que I está ordenado según el inicio del intervalo de llegada de sus arcos $\min(\vec{T}_{e_1}) \leq \cdots \leq \min(\vec{T}_{e_r})$ y O según el inicio del intervalo de partida $\min(T_{f_1}) \leq \cdots \leq \min(T_{f_s})$. Dado un índice $i \in [1 \dots r]$, definimos $next(i)$ como el índice del primer arco en O cuyo intervalo de partida es posterior al final del intervalo de llegada de i, y $weight(i, j)$ como la suma de los valores de $\bar{x}$ para todos los sucesores de e_i que tienen índice al menos j.

$$next(i) = \min\left\{j : \min(T_{f_j}) > \max(\vec{T}_{e_i}) \vee j = s + 1\right\} \tag{3.51}$$

$$weight(i, j) = \sum_{j' \geq j, f_{j'} \in \hat{N}^+(e_i)} \bar{x}_{f_{j'}} \tag{3.52}$$

Ahora, podemos definir la función de programación dinámica f como la violación máxima para k considerando solo los arcos de entrada $i \dots r$ y los arcos de salida $j \dots s$ como

$$f(i,j) = \max\left\{f(i+1,j), f(i+1, \max\{next(i), j\}) + \bar{x}_{e_i} - weight(i,j)\right\}, \tag{3.53}$$

con casos base $f(i,j) = 0$ si $i = r+1$.

Proposición 3.8. *Sea $\hat{\Delta}$ el número máximo de arcos con dependencia temporal incidentes a un vértice. El procedimiento exacto de separación para la TDFI es polinomial y puede resolverse en $\mathcal{O}(|V|\hat{\Delta}^2)$.*

Una demostración detallada de la Proposición 3.8 puede encontrarse en 3.A.1.

Desigualdades Generalized Cut Set con dependencia temporal

En el caso de las TDCSI la separación exacta también es polinomial pero computacionalmente más costosa, lo cual es esperado ya que es una generalización de las TDFI. La idea general es similar a la separación de las desigualdades GCS, en este caso consideramos el dígrafo línea con dependencia temporal $\hat{L}(D)$. Luego, resolvemos un problema de min-cut para cada cliente $k \in V \setminus \{v_s, v_e\}$ donde la capacidad ahora está en los vértices de $\hat{L}(D)$, por lo que necesitamos realizar algunas adaptaciones a este dígrafo resultando en una red con capacidad en los arcos que llamamos D^k_{flow}.

Proposición 3.9. *El procedimiento exacto de separación para el TDCSI es polinomial y puede resolverse en $\mathcal{O}(|V||\hat{A}|^3)$.*

Una prueba detallada de la Proposición 3.9 y la rutina exacta de separación pueden encontrarse en 3.A.1.

3.5. Algoritmo branch-and-cut

En esta sección describimos los ingredientes principales del algoritmo BC.

3.5.1. Técnicas de preprocesamiento

Aplicamos algunas técnicas clásicas de preprocesamiento ampliamente utilizadas para VRP con ventanas de tiempo, incluyendo el acotamiento, la eliminación de arcos, la inferencia de precedencias entre vértices, etc. Debido a limitaciones de espacio omitimos los detalles, pero remitimos al lector a Montero et al. [17] para las ideas clave aplicadas por nuestro algoritmo.

3.5.2. Heurística inicial

Una consideración importante para todo algoritmo branch and bound es disponer de cotas superiores de buena calidad, con la esperanza de reducir el número de nodos explorados en el árbol de búsqueda y lograr un menor tiempo total de ejecución. Nuestro objetivo es encontrar una heurística inicial rápida que proporcione soluciones iniciales ajustadas para alimentar el algoritmo branch and bound.

Consideramos una matheurística basada en la formulación ILP CTTBF (3.15)-(3.25), pero eliminando la dependencia temporal y resolviendo un problema auxiliar independiente del tiempo con un solo período temporal. La motivación detrás de esta estrategia es doble. Primero, como resultado obtenemos una heurística general que puede aplicarse a cualquier problema que pueda ser resuelto por el modelo con dependencia temporal. Segundo, desde el punto de vista de la implementación, solo necesitamos ajustar la definición de los períodos temporales, ya que el resto queda capturado directamente por el modelo. Entonces, la aplicabilidad de esta técnica como heurística depende de dos factores: su tiempo de ejecución y su calidad.

Con respecto al tiempo de ejecución, es fácil ver que el modelo independiente del tiempo tiene considerablemente menos variables y restricciones que su contraparte dependiente del tiempo. Por lo tanto, es posible resolverlo de manera óptima en un par de segundos para muchas de las instancias difíciles dependientes del tiempo. Si no se alcanza la optimalidad en este corto límite de tiempo, a partir de experimentos preliminares se observa que en general produce soluciones factibles de buena calidad. Reconocemos que este enfoque puede tener que ser reconsiderado al intentar resolver instancias más grandes que las consideradas en nuestro caso. El límite de tiempo del heurístico inicial establecido para la experimentación es de dos segundos.

Además, es importante notar que la transformación a una instancia independiente del tiempo puede realizarse de varias maneras. En este trabajo, nuestra elección es preservar la factibilidad. Por lo tanto, la transformación realizada es esencialmente establecer para cada arco $e \in A$ su tiempo de viaje constante e independiente del tiempo c_e como el peor tiempo de viaje durante el horizonte de planificación, definiendo un único período de tiempo con la velocidad de viaje más lenta para e. Cualquier solución factible que utilice estos nuevos tiempos de viaje constantes también es factible en la versión con dependencia temporal, incluso si el problema considera restricciones operativas adicionales, como ventanas de tiempo.

3.5.3. Algoritmo de planos de corte

Otra estrategia efectiva para mejorar la calidad de la cota inferior proporcionada por las relajaciones LP es incorporar desigualdades válidas en un algoritmo de planos de corte y fortalecer la formulación. Como contraparte, esto puede conducir a tiempos de ejecución más lentos para resolver cada relajación. Por lo tanto, la mejora obtenida en la cota inferior, el tiempo necesario para identificar las desigualdades violadas y el tamaño de la formulación resultante (mejorada) requieren un buen compromiso en términos de tiempo de cómputo. Basándonos en experimentos preliminares limitados y teniendo en cuenta el comportamiento reportado en Montero et al. [17], también desarrollamos un algoritmo Cut and Branch (C&B) donde la fase de corte se ejecuta sólo en el nodo raíz. Luego, la información resultante se fija y se resuelve utilizando un solver MILP estándar. Esta estrategia logra tiempos de ejecución más rápidos en comparación con ejecutar una fase de cortes en cada nodo del árbol de enumeración. Por esta razón, también seguimos este enfoque en lo que respecta al algoritmo de planos de corte. Consideramos una estrategia agresiva de planos de corte, donde todas las desigualdades encontradas se añaden a la formulación en cada iteración. La motivación detrás de esta decisión es compensar el hecho de que sólo estamos incluyendo cortes en el nodo raíz.

Las familias de desigualdades GCS, TDFI y TDCSI son válidas para todas las variantes consideradas en este capítulo. Por lo tanto, pueden ser separadas para cualquier variante del CTTBF. Sin embargo, experimentos preliminares muestran que las TDCSI aumentan el tiempo total de ejecución y, por lo tanto, no se incluyen en el algoritmo de planos de corte. Esto puede deberse a que las TDFI capturan la mayor parte de la información con dependencia temporal respecto al flujo en el grafo y los subtours con dependencia temporal podrían ser demasiado específicos. Otra alternativa podría ser mejorar la rutina de separación, con el objetivo de diversificar la estructura de las desigualdades añadidas a la formulación. Observamos que añadir estas desigualdades como cortes puede traducirse en tiempos de cómputo más lentos para resolver la relajación lineal, mientras que aporta una pequeña mejora en la función objetivo. En este sentido, nos enfocamos principalmente en el impacto de las nuevas TDFI combinadas con el modelo compacto. Finalmente, todas las desigualdades GCS de tamaño 2 se incluyen como parte de la formulación, ya que se sabe que tienen un impacto positivo en la relajación LP.

Además, cuando el problema es un TDTSP con ventanas de tiempo, las desigualdades π, σ y (π, σ) también se consideran en el algoritmo de planos de corte. Cabe destacar que las desigualdades (π,σ) se separan únicamente si no se encuentran desigualdades π ni σ violadas.

Con respecto a las rutinas de separación descritas en la Sección 3.4.5, establecemos un umbral de $\Delta = 0{,}1$, lo que significa que una desigualdad debe tener una violación de al menos Δ para ser añadida como corte. Además, solo añadimos cortes mientras la relajación haya mejorado al menos 0,1 en la última iteración de planos de corte. Intuitivamente, esto previene la inclusión de

desigualdades que no tienen un gran impacto en la relajación LP.

3.6. Resultados computacionales

Los experimentos se realizan en una servidor con CPU Intel(R) Core(TM) i5-4570 a 3.20GHz y 16GB de RAM, ejecutando Ubuntu 16.04.4 LTS, y utilizando C++ como lenguaje de programación. Los algoritmos se evalúan en instancias de referencia para tres variantes diferentes del TDESPPRC. Usamos CPLEX 12.8.0 como solver MILP.

Las siguientes secciones muestran los resultados para los tres diferentes problemas considerados, incluyendo los métodos de referencia considerados en cada caso. En este sentido, una comparación directa con algoritmos de última generación para cada variante permite explorar el potencial del algoritmo BC en diferentes contextos. Se reporta un subconjunto de las siguientes métricas para cada experimento, que puede variar dependiendo de la información disponible de los métodos de referencia: el número de instancias resueltas de forma óptima (OPT), el tiempo de ejecución en segundos (time), la brecha en el nodo raíz del BC (%rGap), la cota inferior en el nodo raíz (LB), la cota superior después de alcanzar el límite de tiempo (UB), y el número de nodos (nodes) enumerados en el árbol BC. Se destaca que al reportar el tiempo agregado (time) sobre un conjunto de instancias para diferentes métodos, para fines de comparación se promedia sobre el subconjunto de instancias resueltas por todos los métodos. Además, %rGap se reporta sobre todas las instancias y se calcula usando la Mejor Solución Conocida (BKS, por sus siglas en inglés) para la instancia. A menos que se indique lo contrario, el límite de tiempo de ejecución se establece en una hora.

3.6.1. Instancias TDCESPP y el impacto de la TDFI

En esta sección, nos enfocamos en analizar el impacto de las nuevas restricciones TDFI cuando se usan como cortes. Realizamos nuestros experimentos sobre un nuevo conjunto de instancias de referencia. Los algoritmos se evalúan sobre un subconjunto de las instancias de referencia con 25 clientes propuestas en Dabia et al. [7] para el TDVRPTW, construidas incorporando la dependencia temporal a las conocidas instancias de Solomon, pero descartando las ventanas de tiempo. Este conjunto de datos está compuesto por instancias de pricing que surgieron de diferentes puntos en la ejecución de un algoritmo de generación de columnas sobre las instancias de Dabia et al. [7] sin ventanas de tiempo. Hay seis tipos de instancia, y para cada uno de ellos, cinco beneficios diferentes seleccionados como se explicó anteriormente. Establecemos un límite de tiempo de 2 horas. Los métodos evaluados son los siguientes.

- CTTBF-basic: El algoritmo BC definido en la Sección 3.5, aplicado al problema con capacidades considerando solo las desigualdades GCS.
- CTTBF-TDCESPP: El algoritmo BC definido en la Sección 3.5, adaptado para la variante con capacidades como se describe en la Sección 3.3.4, con todas las desigualdades válidas habilitadas para este escenario (es decir, CPLEX, GCS y TDFI).

			CTTBF-basic				CTTBF-TDCESPP				
Type	Inst	OPT	time	%rGap	%fGap	nodes	OPT	time	%rGap	%fGap	nodes
C1	5	5	146.30	17.56	0.00	25009.00	5	48.76	14.33	0.00	5498.60
R1	5	5	287.17	39.68	0.00	50892.80	5	363.20	38.69	0.00	51703.20
RC1	5	5	588.37	20.78	0.00	55348.80	5	368.10	19.61	0.00	28770.00
C2	5	5	2908.46	12.95	0.00	895246.40	5	786.23	11.03	0.00	171922.60
R2	5	4	53.24	25.44	0.19	9133.00	5	17.50	16.16	0.00	864.00
RC2	5	5	349.65	4.75	0.00	82810.00	5	61.30	2.77	0.00	15008.00
Totals	30	29	745.26	20.19	0.03	192519.55	30	283.03	17.10	0.00	47171.31

Tab. 3.1: Resultados agregados promedio para las instancias TDCESPP.

La Tabla 3.1 presenta los resultados promedio para cada una de las instancias originales. Los resultados más interesantes de la Tabla 3.1 son la reducción promedio del tiempo y también la

disminución en el promedio de nodos enumerados al agregar TDFI como cortes. Además, al considerar las nuevas desigualdades, todas las instancias pueden resolverse de manera óptima dentro del límite de tiempo, mientras que una instancia no pudo demostrarse óptima con la otra configuración. Cabe destacar que estas mejoras se vuelven más significativas al compararlas con las de la formulación no compacta TTBF.

Realizamos un experimento adicional y proporcionamos más evidencia respecto a la estrategia adoptada sobre la ejecución del algoritmo de planos de corte solo en el nodo raíz. Se prueba adicionalmente una versión modificada del algoritmo BC que aplica el algoritmo de planos cortantes en todos los nodos del árbol de enumeración sobre este conjunto de instancias. Los resultados obtenidos indican que esta nueva versión no logra resolver una instancia dentro del límite de tiempo, y que el tiempo promedio de cómputo aumenta a 548.96 segundos, mientras que para el enfoque CB el promedio sobre esas 29 instancias es de 303.11 segundos. Sin embargo, observamos que cualquier variante específica puede beneficiarse de una configuración diferente y personalizada de planos de corte que, además, incluya familias adicionales de desigualdades válidas provenientes de problemas relacionados (independientes del tiempo) y que pueda aplicar un esquema distinto en los nodos internos. Dado que el objetivo de este capítulo es evaluar el enfoque en una amplia gama de problemas, fijamos la configuración descrita en la Sección 3.5 para el algoritmo BC.

3.6.2. TDOP sin ventanas de tiempo

Otra variante que consideramos es el TDOP sin ventanas de tiempo estudiado en Verbeeck et al. [25], donde la función objetivo maximiza la ganancia recolectada por los vértices visitados y existe una restricción adicional que limita la ruta a una duración máxima de $t_{\max}$.

Con respecto a los experimentos, consideramos los Conjuntos 1, 2 y 3 propuestos en Verbeeck et al. [25], que tienen instancias con 32, 21 y 33 vértices, respectivamente. Las instancias consideradas en los conjuntos restantes tienen más de 60 vértices, lo que resulta prohibitivo hasta ahora para enfoques exactos. Para cada instancia, establecemos un límite de tiempo de una hora para el tiempo máximo de ejecución.

Los métodos evaluados son los siguientes:

- V-TTBF: Los resultados reportados para la formulación MILP en Verbeeck et al. [25]. Cabe destacar que esta formulación es similar al TTBF estándar e incluye el fortalecimiento análogo a (3.21). No reportan ninguna familia de desigualdades añadidas como cortes.
- CTTBF-basic: El CTTBF descrito en la Sección 3.3, incluyendo GCS para $|S| = 2$ como parte de la formulación.
- CTTBF-TDOP: El algoritmo BC descrito en la Sección 3.5, adaptado para el TDOP.

Reportamos en la Tabla 3.2 los resultados agregados por conjunto de instancias donde se impone una duración máxima $t_{\max}$ al makespan de la ruta, es decir, teniendo $t_{v_s} = 0$. La clave de la tabla es la misma que en la sección anterior. Los resultados detallados pueden consultarse en el material suplementario que acompaña a este capítulo. El primer mensaje de la tabla es que el CTTBF reduce significativamente los tiempos de cómputo en las dos variantes consideradas, resolviendo 5 instancias más que el enfoque estándar al ajustar el límite de tiempo a 5400 segundos.[1]. En cuanto al impacto del TDFI, hay una reducción en el número de nodos enumerados, pero esto no se traduce directamente en los tiempos de cómputo. Aunque cierran el mismo número total de instancias, CTTBF y CTTBF-TDOP difieren en las instancias particulares resueltas dentro del límite de tiempo. A partir de los experimentos detallados, notamos que el BC es muy sensible a la calidad de la solución inicial. Por lo tanto, realizamos un experimento adicional con los mismos métodos pero estableciendo el BKS del experimento previo como solución inicial, y los resultados se reportan en la Tabla 3.3. Esto puede verse como una emulación de la integración entre la heurística propuesta en Verbeeck et al. [25] y nuestro enfoque. Los tiempos de cómputo se reducen

[1] Basándonos en la información limitada proporcionada por Verbeeck et al. [25], estimamos que nuestra computadora es aproximadamente 1.5 veces más rápida.

		V-TTBF		CTTBF-basic					CTTBF-TDOP				
Tipo	Inst	OPT	time	OPT	time	%rGap	%fGap	nodes	OPT	time	%rGap	%fGap	nodes
1	9	8	6576.86	7	98.01	6.16	0.35	10472.57	8	111.48	5.94	0.19	7939.29
2	9	9	2319.11	9	40.29	7.43	0.00	7111.56	9	36.68	7.98	0.00	3222.33
3	9	7	2933.71	8	25.35	6.44	0.15	2563.71	7	40.29	6.30	0.38	3468.43
Totales	27	24	3802.00	24	53.31	6.68	0.17	6750.35	24	60.54	6.74	0.19	4732.83

Tab. 3.2: Resultados promedio agregados para instancias TDOP con restricción de makespan.

		V-TTBF		CTTBF-basic (BKS)					CTTBF-TDOP (BKS)				
Tipo	Inst	OPT	time	OPT	time	%rGap	%fGap	nodes	OPT	time	%rGap	%fGap	nodes
1	9	8	6576.86	7	12.22	3.65	0.38	2778.43	8	12.74	3.42	0.19	1686.43
2	9	9	2319.11	9	15.61	7.62	0.00	2305.44	9	19.55	5.85	0.00	1543.11
3	9	7	2933.71	9	5.25	3.50	0.00	400.29	8	5.27	3.33	0.14	260.00
Totales	27	24	3802.00	25	11.43	4.93	0.13	1869.57	25	13.13	4.20	0.11	1196.22

Tab. 3.3: Resultados agregados promedio para instancias TDOP con restricción de makespan usando BKS como solución inicial.

en alrededor de un 20 % en promedio. Cabe destacar que considerando ambos experimentos somos capaces de probar la optimalidad para dos nuevas instancias, 3.i y 3.h.

Alternativamente, se puede considerar que la restricción sobre el tiempo de la ruta afecta la duración en lugar del makespan, pudiendo demorar la partida si es conveniente. Según nuestro conocimiento, esta variante no ha sido considerada previamente en la literatura sobre dependencia temporal. Realizamos experimentos adicionales con las instancias de la versión anterior simplemente eliminando la restricción $t_{v_s} = 0$. Los resultados para CTTBF-basic y CTTBF-TDOP se reportan en la Tabla 3.4, mientras que los resultados obtenidos comenzando desde el BKS se presentan en la Tabla 3.5. Estos están alineados con los hallazgos previos. Aunque a primera vista el problema parece ser más difícil de resolver que su contraparte con makespan, la inclusión del TDFI reduce el número de nodos enumerados en el árbol. Comenzar desde el BKS, nuevamente, mejora los tiempos de cómputo en general. Destacamos que el TDFI muestra un impacto positivo en las instancias del Conjunto 1, que parecen ser las más desafiantes.

3.6.3. TDPTP con ventanas de tiempo y recolección y entrega

A continuación evaluamos el desempeño de nuestro modelo en la variante TDPTP que considera recolecciones y entregas, ventanas de tiempo y capacidades. Los experimentos se realizan sobre un subconjunto de las instancias propuestas en Sun et al. [21]. Primero, destacamos que la formulación MILP propuesta en Sun et al. [21] inspiró algunos trabajos preliminares sobre algoritmos BC para otras variantes de problemas de un solo vehículo con dependencia temporal, incluyendo este capítulo. Segundo, notamos que esta formulación MILP no es el mejor enfoque reportado en Sun et al. [21]. De hecho, ellos desarrollan un algoritmo basado en labeling que supera al método anterior, el cual explota las ventanas de tiempo estrechas de las instancias. En este sentido, nuestro objetivo es mostrar las mejoras en el desempeño al usar el CTTBF junto con las nuevas desigualdades válidas. Con respecto a los experimentos, establecemos un límite de tiempo de una hora para el tiempo de ejecución.

- ORIG-TTBF: El algoritmo BC introducido en la Sección 3 de Sun et al. [21] que es una versión más débil de CTTBF ya que algunas de sus restricciones han sido reforzadas. No reportan ninguna familia de desigualdades añadidas como cortes a la formulación.

			CTTBF-basic				CTTBF-TDOP				
Type	Inst	OPT	time	%rGap	%fGap	nodes	OPT	time	%rGap	%fGap	nodes
1	9	7	993.36	7.06	0.20	69577.63	7	978.16	6.36	0.20	42630.88
2	9	9	12.87	7.65	0.00	1957.78	9	11.69	6.44	0.00	1574.33
3	9	9	85.91	1.50	0.00	4998.56	9	107.98	1.49	0.13	4262.89
Totals	27	25	209.43	5.40	0.07	20473.04	25	212.09	4.77	0.11	12384.44

Tab. 3.4: Resultados promedio agregados para instancias TDOP con restricción de duración.

			CTTBF-basic (BKS)				CTTBF-TDOP (BKS)				
Type	Inst	OPT	time	%rGap	%fGap	nodes	OPT	time	%rGap	%fGap	nodes
1	9	8	65.37	6.48	0.20	8233.50	8	40.21	4.64	0.20	4068.00
2	9	9	2.10	0.94	0.00	3.67	9	3.29	2.35	0.00	29.11
3	9	9	2.50	0.00	0.00	0.00	9	2.61	0.00	0.00	0.00
Totals	27	26	21.71	2.47	0.07	2534.65	26	14.41	2.33	0.07	1261.77

Tab. 3.5: Resultados agregados promedio para instancias TDOP con restricción de duración usando BKS como solución inicial.

- CTTBF-basic: El CTTBF descrito en la Sección 3.3 adaptado para esta variante del TDPTP, incluyendo GCS para $|S| = 2$ como parte de la formulación y un plano de corte considerando los GCS restantes en el nodo raíz.
- CTTBF-CTWPD: El algoritmo BC definido en la Sección 3.5 adaptado para la variante con capacidades con recolecciones y entregas y ventanas de tiempo según lo especificado en la Sección 3.3.4, con todos los cortes válidos para este problema habilitados (es decir, CPLEX, GCS, TDFI).
- SUN-L: los resultados obtenidos por el algoritmo de labeling personalizado propuesto en Sun et al. [21].

Una primera mirada a los resultados indica que CTTBF-CTWPD es capaz de resolver una instancia más que ORIG-TTBF y CTTBF-basic, aunque obtiene una reducción drástica en los tiempos de cómputo requeridos. Señalamos, sin embargo, que la computadora utilizada en Sun et al. [21] es considerablemente más rápida que la nuestra, y que ellos establecieron un límite de tiempo de 24 horas mientras que nuestro método se detiene después de solo 1 hora. Por lo tanto, incluso al ignorar la diferencia en el entorno y considerando las instancias resueltas dentro de 1 hora en ambos casos, CTTBF-CTWPD es capaz de resolver 4 instancias más en mucho menos tiempo total de cómputo. Esto demuestra que tanto las mejoras en la formulación combinadas con el algoritmo BC a medida son efectivas para mejorar el comportamiento general. Sin embargo, destacamos que el enfoque de labeling supera a los algoritmos basados en MILP en estas instancias.

3.6.4. TDTSPTW

A continuación analizamos el desempeño de nuestro algoritmo en la variante TDTSPTW. Incluimos dos conjuntos de instancias, Set1 y w100, propuestos originalmente en Arigliano et al. [3] y posteriormente considerados en Montero et al. [17] y Vu et al. [28]. La diferencia entre ambos conjuntos de datos es el tamaño de la partición del horizonte de planificación en períodos de tiempo, donde el primero tiene considerablemente menos períodos que el segundo. Debido a algunos problemas numéricos detectados, para este experimento en particular la tolerancia del mip gap se establece en $1e^{-6}$. Los métodos evaluados son los siguientes:

- DDD-TD-TSPTW: El algoritmo introducido en Vu et al. [28] que considera un modelo con una discretización del horizonte temporal que se refina iterativamente hasta encontrar un óptimo. Su algoritmo original funciona para el problema de minimización del makespan, pero también se proporcionó una adaptación para resolver la versión de minimización de la duración.
- TTBF-CB: El algoritmo para el TDTSPTW propuesto en Montero et al. [17]. Consideran el modelo original que tiene una variable t_{ij}^m para cada $ij \in A, m \in T^{ij}$, e incluyen además las desigualdades GCS de tamaño dos explícitamente en la formulación. Además, varias familias de desigualdades como SECs, π, σ y (π, σ), se añaden dinámicamente como cortes en el nodo raíz.
- CTTBF-basic: La formulación CTTBF, incluyendo las GCS de tamaño dos explícitamente en la formulación así como las desigualdades válidas del TSPTW independiente del tiempo (SECs, π, σ y (π, σ)) añadidas dinámicamente como cortes en el nodo raíz.

	ORIG-TTBF			CTTBF-basic					CTTBF-CTWPD					SUN-L		
Inst	time	UB	%fGap	time	UB	%rGap	%fGap	nodes	time	UB	%rGap	%fGap	nodes	time	UB	%fGap
AA10	1.95	27.14	0.00	0.08	27.14	0.00	0.00	0	0.07	27.14	0.00	0.00	0	0.23	27.14	0.00
AA15	4.66	37.77	0.00	1.15	37.77	30.95	0.00	27	1.74	37.77	29.48	0.00	41	0.37	37.77	0.00
AA20	24.93	73.70	0.00	8.26	73.7	46.35	0.00	498	8.38	73.7	44.59	0.00	574	0.80	73.70	0.00
AA25	46344.06	203.01	0.00	-	233.47	39.27	13.05	118224	1256.21	203.01	38.96	0.00	47923	1.59	203.01	0.00
AA30	-	543.48	50.92	-	327.8	39.49	18.63	55515	-	327.15	38.25	18.47	54447	2.12	266.72	0.00
BB10	1.45	39.26	0.00	0.34	39.26	6.75	0.00	5	0.28	39.26	13.10	0.00	2	0.22	39.26	0.00
BB15	121.03	99.58	0.00	3.71	99.58	26.92	0.00	294	4.92	99.58	28.96	0.00	532	0.50	99.58	0.00
BB20	1669.74	138.05	0.00	257.74	138.05	43.81	0.00	20439	110.13	138.05	42.32	0.00	7400	1.95	138.05	0.00
BB25	48616.51	147.88	0.00	1078.01	147.88	50.96	0.00	48851	1221.85	147.88	49.39	0.00	57455	2.25	147.88	0.00
BB30	-	515.40	46.75	-	387.75	46.52	29.22	62668	-	394.17	44.42	30.37	51803	2.36	274.45	0.00
CC10	3.29	57.63	0.00	0.64	57.63	16.17	0.00	36	0.71	57.63	16.21	0.00	40	0.27	57.63	0.00
CC15	44.34	98.29	0.00	5.86	98.29	28.68	0.00	904	7.04	98.29	29.03	0.00	1092	0.63	98.29	0.00
CC20	7935.29	97.27	0.00	62.01	97.27	53.85	0.00	4204	63.42	97.27	54.07	0.00	3988	1.25	97.27	0.00
CC25	-	1316.69	82.82	-	505.73	58.33	55.27	35098	-	504.05	58.03	55.12	29665	11.95	226.23	0.00
CC30	-	1674.74	81.11	-	683.07	56.74	53.68	23688	-	680.9	55.94	53.53	19384	27.64	316.39	0.00
DD10	1.44	85.88	0.00	0.18	85.88	0.00	0.00	0	0.18	85.88	9.04	0.00	9	0.25	85.88	0.00
DD15	54.65	99.05	0.00	2.96	99.05	11.01	0.00	199	2.20	99.05	10.54	0.00	111	0.61	99.05	0.00
DD20	4959.48	182.14	0.00	220.93	182.14	28.70	0.00	14003	342.65	182.16	27.99	0.00	20907	2.11	182.14	0.00
DD25	-	1419.81	82.38	-	521.89	55.82	52.07	27790	-	515.25	55.20	51.45	25033	199.74	250.16	0.00
DD30	-	1733.62	80.18	-	691.17	53.22	50.28	15005	-	678.07	52.31	49.32	12634	80.42	343.65	0.00
Totals	4879.90		21.21	126.30		34.68	13.61	6881.54	135.66		34.89	12.91	7088.54	0.88		0.00

Tab. 3.6: Resultados para instancias del TDPTP con ventanas de tiempo y recolecciones y entregas.

	Conjunto 1 (952 instancias)					w100 (960 instancias)				
Método	OPT	time	%rGap	%fGap	nodes	OPT	time	%rGap	%fGap	nodes
DDD-TD-TSPTW	952	10.94	-	-	-	960	1.22	-	-	-
TTBF-CB	940	26.22	43.56	0.02	3848.83	214	96.62	40.42	0.03	11217.11
CTTBF-basic	951	8.28	2.85	0.00	1039.77	958	20.70	1.03	0.00	2448.85
CTTBF-TDTSPTW	950	7.36	2.61	0.00	823.76	960	21.39	0.58	0.00	1731.02

Tab. 3.7: Resultados promedio agregados para instancias de TDTSPTW con objetivo de makespan.

	Conjunto 1 (952 instancias)					w100 (960 instancias)				
Método	OPT	time	%rGap	%fGap	nodes	OPT	time	%rGap	%fGap	nodes
DDD-TD-TSPTW	930	143.45	-	-	-	955	106.2	-	-	-
CTTBF-basic	945	13.72	4.23	0.02	1977.21	950	44.85	1.73	0.00	4793.87
CTTBF-TDTSPTW	943	12.87	3.85	0.03	1512.88	957	45.64	0.92	0.00	2123.83

Tab. 3.8: Resultados agregados promedio para instancias de TDTSPTW con objetivo de duración.

- CTTBF-TDTSPTW: El algoritmo BC definido en la Sección 3.5 adaptado para la variante TSP con ventanas de tiempo considerando todos los cortes válidos para este problema (es decir, GCS, π, σ, (π, σ), TDFI).

La Tabla 3.7 presenta los resultados promedio para cada uno de los conjuntos de datos con el objetivo de minimizar el makespan. De manera similar, en la Tabla 3.8 presentamos los resultados promedio para cada uno de los conjuntos de datos con el objetivo de minimizar la duración. En este caso solo comparamos los métodos DDD-TD-TSPTW y CTTBF-TSPTW. Agregamos los resultados para que coincidan con el formato presentado en Vu et al. [28].

El mensaje principal en la Tabla 3.7 es que la relajación fortalecida de CTTBF-TDTSPTW, en combinación con el modelo compacto y los cortes TDFI, muestra una mejora significativa sobre los resultados obtenidos en Montero et al. [17]. En particular, observamos que el promedio de %rGap pasó de aproximadamente 40 % en TTBF-CB a menos de 3 % en la nueva formulación. Además, esto se traduce en tiempos de cómputo más cortos, ya que solo una instancia no es resuelta por ninguna de las alternativas CTTBF frente a las 12 instancias no resueltas por TTBF-CB, requiriendo considerablemente menos tiempo total.

Sin embargo, cuando el objetivo implica encontrar la ruta con el makespan mínimo, nuestro algoritmo resuelve una instancia menos que DDD-TD-TSPTW, aunque los tiempos promedio de ejecución son similares e incluso menores para el Conjunto 1. En contraste, cuando la función objetivo minimiza la duración, nuestro método supera a DDD-TD-TSPTW. Notar que podemos resolver 15 instancias más con este objetivo.[2] Además, nuestro algoritmo requiere menos del 20 % del tiempo al minimizar el makespan y alrededor del 40 % para la duración. Creemos que estos resultados muestran la versatilidad del modelo y lo bien que se adapta a diferentes objetivos.

3.6.5. Impacto de las restricciones TDCSI

Finalmente, presentamos algunos resultados para ofrecer una visión sobre los beneficios potenciales del TDCSI. Como se mencionó anteriormente, no se han incluido directamente en el algoritmo BC porque la rutina de separación tiende a ralentizar la ejecución general del algoritmo. Las TDFI, generalizadas por las TDCSI, pueden verse como un compromiso entre la mejora en la cota inferior al agregar cortes y la eficiencia de la rutina de separación. Realizamos un experimento adicional y específico para mostrar que las TDCSI mejoran aún más la calidad de la cota inferior.

Consideramos todas las instancias de referencia de las seis variantes estudiadas previamente para evaluar las mejoras en la cota inferior en el nodo raíz del algoritmo BC utilizando los siguientes métodos:

- CTTBF-basic: la formulación CTTBF de la Sección 3.3, con las GCS como planos de corte. Para el TDTSPTW, también imitamos el valor de la cota inferior del modelo considerado en

[2] También destacamos que nuestro modelo puede manejar tiempo continuo mientras que DDD-TD-TSPTW requiere una discretización al minimizar la duración.

Montero et al. [17], al que nos referimos como TTBF, e incluimos además las restricciones de precedencia π, σ y (π, σ) como planos de corte.

- CTTBF-TDFI: CTTBF-basic incluyendo también el TDFI en el algoritmo de planos de corte.
- CTTBF-TDCSI: CTTBF-basic incluyendo también el TDCSI en el algoritmo de planos de corte.

Todos los experimentos consideran únicamente el nodo raíz, por lo que no se realiza enumeración. El algoritmo de planos de corte ha sido modificado y no se requiere una mejora mínima en el valor del LP. Por lo tanto, no limitamos el número de rondas en el algoritmo de planos de corte. Además, utilizamos la BKS como solución inicial y establecemos un límite de tiempo de una hora por instancia (excepto para el TDTSPTW donde, debido al gran número de instancias, este límite se reduce a tres minutos por instancia). Estos cambios pueden afectar el comportamiento del algoritmo de planos de corte así como algunos procedimientos estándar por defecto del solver, causando algunas diferencias en la cota inferior final alcanzada. Los resultados se reportan en las tablas 3.9 - 3.14, con diferentes niveles de agregación según la variante. La tabla reporta el tiempo promedio de ejecución sobre todas las instancias, el promedio de %rGap y el número de cortes añadidos (#cuts).

	CTTBF-basic			CTTBF-TDFI			CTTBF-TDCSI		
class	time	%rGap	#cuts	time	%rGap	#cuts	time	%rGap	#cuts
C1	7.53	17.50	63.80	9.45	15.01	169.40	388.06	9.24	2864.60
R1	7.86	39.08	25.00	8.63	38.64	67.80	28.80	36.42	575.00
RC1	13.21	20.77	154.40	15.81	19.77	235.60	147.92	13.40	1308.20
C2	7.27	12.99	95.80	10.34	11.03	175.20	1997.70	7.16	8059.40
R2	5.64	24.14	50.00	6.75	0.53	161.60	739.30	0.25	3540.60
RC2	8.59	4.76	137.80	8.36	1.35	233.00	783.25	0.03	3879.60
Totals	8.35	19.87	87.80	9.89	14.39	173.77	680.84	11.08	3371.23

Tab. 3.9: Resultados en instancias TDCESPP limitados al nodo raíz.

	CTTBF-basic			CTTBF-TDFI			CTTBF-TDCSI		
clase	time	%rGap	#cuts	time	%rGap	#cuts	time	%rGap	#cuts
1	10.18	3.74	66.11	44.03	3.46	158.44	825.52	0.60	2247.00
2	3.11	9.87	30.22	3.88	7.43	77.11	6.31	5.49	265.78
3	7.05	2.82	51.44	69.96	2.73	140.89	805.53	2.11	935.11
Totales	6.78	5.48	49.26	39.29	4.54	125.48	545.79	2.73	1149.30

Tab. 3.10: Resultados en instancias TDOP con restricción de makespan limitada al nodo raíz.

	CTTBF-basic			CTTBF-TDFI			CTTBF-TDCSI		
class	time	%rGap	#cuts	time	%rGap	#cuts	time	%rGap	#cuts
1	10.86	5.83	66.33	11.89	4.65	164.44	554.29	1.60	2870.22
2	2.22	3.84	27.78	2.97	2.66	66.11	2.60	2.40	126.78
3	2.53	0.00	37.56	2.73	0.00	70.67	2.74	0.00	91.11
Totals	5.20	3.22	43.89	5.86	2.44	100.41	186.54	1.33	1029.37

Tab. 3.11: Resultados en instancias TDOP con restricción de duración limitada al nodo raíz.

Podemos observar reducciones en la %rGap en todas las variantes, aunque de diferente magnitud según el caso. Las más significativas se obtienen para el TDCESPP y el TDOP, y respecto a las variantes que incluyen ventanas de tiempo, las mejoras son moderadas. En todos los casos, notamos un incremento significativo en el número total de cortes añadidos para CTTBF-TDCSI. Destacamos que para las dos variantes del TDTSPTW hay un número significativo de instancias que alcanzan el límite de tiempo, lo que también afecta los resultados finales. En general, estos resultados sugieren que existe margen para mejoras adicionales desarrollando rutinas de separación más eficientes para el TDCSI e incluyendo un conjunto más amplio de estas restricciones en un algoritmo BC.

	CTTBF-basic			CTTBF-TDFI			CTTBF-TDCSI		
class	time	%rGap	#cuts	time	%rGap	#cuts	time	%rGap	#cuts
10	0.18	3.87	11.25	0.18	3.94	14.00	0.19	3.83	37.00
15	0.86	24.43	41.00	1.03	24.32	55.50	3.75	23.15	418.75
20	2.03	43.09	38.00	2.24	41.51	64.50	3.31	41.04	326.25
25	5.13	51.60	69.50	5.58	50.22	129.25	266.30	50.53	3507.00
30	9.45	48.78	49.50	10.47	47.81	106.25	18.97	47.54	535.25
Totals	3.53	34.35	41.85	3.90	33.56	73.90	58.50	33.22	964.85

Tab. 3.12: Resultados en instancias TDPTP con ventanas de tiempo y recolección y entrega limitadas al nodo raíz.

	TTBF			CTTBF-basic			CTTBF-TDFI			CTTBF-TDCSI		
clase	time	%rGap	#cuts	time	%rGap	#cuts	time	%rGap	#cuts	time	%rGap	#cuts
set1-15	0.03	5.29	11.74	0.03	2.87	7.62	0.04	2.40	14.99	1.54	1.97	181.76
set1-20	0.06	5.07	21.64	0.06	2.63	13.72	0.08	2.35	24.43	5.01	2.13	310.91
set1-30	0.15	5.88	49.13	0.15	2.60	27.65	0.19	2.46	44.00	1.35	2.41	280.12
set1-40	0.38	5.62	91.27	0.41	2.42	53.27	0.47	2.38	81.54	5.42	2.36	489.82
w100-15	0.11	14.22	9.75	0.11	0.86	3.49	0.31	0.29	45.76	8.86	0.25	392.00
w100-20	0.17	23.10	17.26	0.20	0.97	5.39	0.55	0.37	73.40	17.39	0.28	701.79
w100-30	0.35	28.52	38.84	0.41	0.72	14.00	2.13	0.36	160.45	50.37	0.36	1562.31
w100-40	0.50	17.13	55.83	0.77	0.68	23.52	4.81	0.50	208.53	44.48	0.54	1122.28
Totales	0.22	13.14	36.94	0.27	1.71	18.57	1.08	1.38	81.83	16.86	1.28	631.47

Tab. 3.13: Resultados en instancias TDTSPTW con objetivo de makespan limitado al nodo raíz.

3.7. Conclusiones

En este capítulo proponemos una nueva formulación MILP para el TDPTPRC y desarrollamos un algoritmo BC a medida que, con algunas adaptaciones, puede aplicarse a diferentes problemas. Desde un punto de vista teórico, proporcionamos una garantía de que el procedimiento aplicado para obtener la formulación CTTBF partiendo de una versión fortalecida de la TTBF propuesta en Sun et al. [21] mediante la reducción de variables no afecta la calidad de la relajación LP. Además, proponemos varias familias de desigualdades válidas que explotan la dependencia temporal de la función de tiempo de viaje. Estas familias se estudian en detalle y desarrollamos algoritmos de separación eficientes para incluirlas en un algoritmo BC. El algoritmo propuesto se evalúa en cuatro problemas diferentes, incluyendo dos benchmarks recientes de la literatura relacionada. Cabe destacar que nuestros algoritmos demostraron ser bastante efectivos en todos los casos, especialmente para dos variantes diferentes del TDTSPTW. Como investigación futura, consideramos que vale la pena investigar más familias que exploten la dependencia temporal, en particular basándose en el TDCSI. Además, se pueden obtener mejoras significativas encontrando planos de corte que conecten la dependencia temporal con restricciones específicas de recursos, como las ventanas de tiempo. Desde una perspectiva práctica, el enfoque propuesto podría ser adecuado para escenarios donde no hay recursos limitantes disponibles o, por ejemplo, en presencia de ventanas de tiempo blandas.

Referencias

[1] Hernán Abeledo et al. «The time dependent traveling salesman problem: polyhedra and algorithm». En: *Mathematical Programming Computation* 5.1 (2012), págs. 27-55.

[2] Anna Arigliano et al. «A branch-and-bound algorithm for the time-dependent travelling salesman problem». En: *Networks* 72.3 (2018), págs. 382-392.

[3] Anna Arigliano et al. *Time Dependent Traveling Salesman Problem with Time Windows: Properties and an Exact Algorithm*. Inf. téc. 2015.

[4] Egon Balas et al. «The precedence-constrained asymmetric traveling salesman polytope». En: *Math. Program.* 68 (3 1995), págs. 241-265. ISSN: 0025-5610.

[5] Roberto Baldacci et al. «New route relaxation and pricing strategies for the vehicle routing problem». En: *Oper. Res.* 59.5 (2011), págs. 1269-1283.

	TTBF			CTTBF-basic			CTTBF-TDFI			CTTBF-TDCSI		
class	time	%rGap	#cuts	time	%rGap	#cuts	time	%rGap	#cuts	time	%rGap	#cuts
set1-15	0.05	9.50	22.61	0.04	4.48	12.63	0.06	3.78	22.03	3.20	3.34	263.44
set1-20	0.09	8.62	38.52	0.09	4.04	20.61	0.12	3.63	35.40	4.87	3.42	324.36
set1-30	0.19	8.10	65.98	0.20	3.47	35.97	0.25	3.30	59.84	2.06	3.28	366.82
set1-40	0.47	8.51	122.26	0.57	3.98	69.90	0.68	3.88	106.49	9.11	3.86	698.82
w100-15	0.15	31.10	17.11	0.15	2.41	4.80	0.39	1.05	83.79	19.66	0.97	908.51
w100-20	0.25	34.09	28.37	0.28	1.74	6.55	0.97	0.77	116.00	35.97	0.65	1374.27
w100-30	0.39	42.26	49.23	0.52	1.37	20.38	3.23	0.76	208.35	79.05	0.80	2213.83
w100-40	0.61	23.62	84.78	0.79	1.36	29.81	6.25	0.89	265.03	69.23	1.02	1759.05
Average	0.28	20.77	53.61	0.33	2.85	25.06	1.50	2.25	112.38	27.99	2.16	991.21

Tab. 3.14: Resultados en instancias de TDTSPTW con objetivo de duración limitado al nodo raíz.

[6] Jean-François Cordeau et al. «Analysis and Branch-and-Cut Algorithm for the Time-Dependent Travelling Salesman Problem». En: *Transport. Sci.* 48.1 (2014), págs. 46-58.

[7] Said Dabia et al. «Branch and Price for the Time-Dependent Vehicle Routing Problem with Time Windows». En: *Transportation Science* 47.3 (2013), págs. 380-396.

[8] Michael Drexl. «A note on the separation of subtour elimination constraints in elementary shortest path problems». En: *European Journal of Operational Research* 229.3 (2013), págs. 595-598.

[9] Michael Drexl et al. «Solving elementary shortest-path problems as mixed-integer programs». En: *OR Spectrum* 36.2 (2014), págs. 281-296. ISSN: 14366304.

[10] Dominique Feillet. «A tutorial on column generation and branch-and-price for vehicle routing problems». En: *4OR* 8.4 (2010), págs. 407-424.

[11] Michel Gendreau et al. «Time-dependent routing problems: a review». En: *Comput. Oper. Res.* 64 (2015), págs. 189-197.

[12] Yixiao Huang et al. «Time-dependent vehicle routing problem with path flexibility». En: *Transp. Res. B* 95 (2017), págs. 169-195.

[13] Soumia Ichoua et al. «Vehicle dispatching with time-dependent travel times». En: *Eur. J. Oper. Res.* 144.2 (2003), págs. 379-396.

[14] Mads Kehlet Jepsen et al. «A branch-and-cut algorithm for the capacitated profitable tour problem». En: *Discrete Optimization* 14 (2014), págs. 78-96.

[15] Gonzalo Lera-Romero et al. «Linear edge costs and labeling algorithms: The case of the time-dependent vehicle routing problem with time windows». En: *Networks* 76.1 (2020), págs. 24-53.

[16] Juan José Miranda-Bront et al. «Facets and valid inequalities for the time-dependent travelling salesman problem». En: *Eur. J. Oper. Res.* 236.3 (2014), págs. 891-902.

[17] Agustín Montero et al. «An integer programming approach for the time-dependent traveling salesman problem with time windows». En: *Comput. Oper. Res.* 88 (2017), págs. 280-289.

[18] Jean-Claude Picard et al. «The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling». En: *Operations Research* 26.1 (1978), págs. 86-110.

[19] Remy Spliet et al. «The Time Window Assignment Vehicle Routing Problem with Time-Dependent Travel Times». En: *Transp. Sci.* 52.2 (2018), págs. 261-276.

[20] Peng Sun et al. «The time-dependent capacitated profitable tour problem with time windows and precedence constraints». En: *Eur. J. Oper. Res.* 264.3 (2018), págs. 1058-1073.

[21] Peng Sun et al. «The time-dependent capacitated profitable tour problem with time windows and precedence constraints». En: *Eur. J. Oper. Res.* 264.3 (2018), págs. 1058-1073.

[22] Peng Sun et al. «The time-dependent pickup and delivery problem with time windows». En: *Transport. Res. B-Meth.* 116 (2018), págs. 1-24.

[23] Leonardo Taccari. «Integer programming formulations for the elementary shortest path problem». En: *European Journal of Operational Research* 252.1 (2016), págs. 122-130. ISSN: 03772217.

[24] Paolo Toth et al., eds. *Vehicle Routing: Problem, Methods and Applications*. Philadelphia, PA, USA: Society for Industrial y Applied Mathematics, 2014. ISBN: 0-89871-498-2.

[25] Cédric Verbeeck et al. «A fast solution method for the time-dependent orienteering problem». En: *European Journal of Operational Research* 236.2 (2014), págs. 419-432.

[26] Cédric Verbeeck et al. «The time-dependent orienteering problem with time windows: a fast ant colony system». En: *Annals OR* 254.1-2 (2017), págs. 481-505.

[27] Duc Minh Vu et al. «Dynamic Discretization Discovery for Solving the Time-Dependent Traveling Salesman Problem with Time Windows». En: *Transport. Sci.* 54.3 (2020), págs. 703-720.

[28] Duc Minh Vu et al. *Solving Time Dependent Traveling Salesman Problems with Time Windows*. Inf. téc. 2018, págs. 1-30.

3.A. Apéndice

3.A.1. Rutinas de separación

Demostración de la Proposición 3.8

A continuación demostramos que la violación máxima de una desigualdad (3.39) es exactamente $f(1,1)$. El caso cuando $i = r+1$ se satisface trivialmente ya que no hay arcos entrantes a k. Sea $i < r+1$, F^* el conjunto de arcos que entran a k y producen la TDFI más violada y $S^* = \hat{N}^+(F^*)$. Tenemos dos casos: o bien $e_i \in F^*$ o $e_i \notin F^*$. Si $e_i \notin F^*$ entonces la desigualdad más violada es la misma que la producida por el subproblema $(i+1, j)$, es decir, considerando solo los arcos $e_{i+1} \cdots e_r$. El caso interesante es cuando $e_i \in F^*$, aquí sabemos que la violación de la desigualdad (3.39) definida por F^* es $\bar{x}_{e_i} - weight(i,j)$ más la violación máxima producida por $F' = F^* \setminus \{e_i\}$ y los sucesores $S' = S^* \setminus \hat{N}^+(e_i)$. A continuación procedemos a mostrar que S' contiene solo arcos en O comenzando desde el índice $\max\{next(i), j\}$, lo cual completa la demostración.

Supongamos que existe $i' > i, j' < \max\{next(i), j\}$ tal que $f_{j'}$ es un sucesor de $e_{i'}$ pero no es un sucesor de e_i. Por definición, $\vec{T}_{e_{i'}} \cap T_{f_{j'}} \neq \varnothing$, entonces $\max(T_{f_{j'}}) \geq \min(\vec{T}_{e_{i'}})$. Como $j' < next(i)$ y O está ordenado, también sabemos que $\min(T_{f_{j'}}) \leq \max(\vec{T}_{e_i})$. Finalmente, dado que I está ordenado, sabemos que $\min(\vec{T}_{e_i}) \leq \min(\vec{T}_{e_{i'}})$. Por lo tanto, se cumple que $\max(T_{f_{j'}}) \geq \min(\vec{T}_{e_i}) \geq \min(T_{f_{j'}})$, por lo que $f_{j'}$ era un sucesor de e_i, lo cual es una contradicción.

Hemos probado todos los casos, por lo tanto el algoritmo debe ser correcto. Nótese que podemos reconstruir fácilmente un conjunto $F \subseteq \hat{A}$ que produce una desigualdad (3.39) con violación $f(0,0)$ utilizando la decisión tomada en cada paso recursivo.

Finalmente, discutimos la complejidad del algoritmo. Sea $\hat{\Delta}$ el número máximo de arcos con dependencia temporal incidentes a un vértice k. El preprocesamiento de las funciones $weight(i,j)$, $next(i)$ para cada i, j se puede realizar en tiempo $\mathcal{O}(\hat{\Delta}^2)$. Luego, cada paso de la función f puede calcularse en tiempo constante, por lo que si la memoización se implementa usando una matriz, todos los posibles estados de f pueden calcularse en tiempo $\mathcal{O}(\hat{\Delta}^2)$. La reconstrucción del corte después de calcular todos los estados para f se puede hacer en tiempo lineal respecto a $\hat{\Delta}$, por lo que la complejidad total de la rutina de separación es $\mathcal{O}(\hat{\Delta}^2)$ para cada vértice $k \in V$. □

Demostración de la Proposición 3.9

Para la rutina de separación adaptamos el dígrafo $\hat{L}(D)$ utilizando técnicas comunes de flujo máximo para construir una red con capacidades de arco D^k_{flow} para cada cliente $k \in V \setminus \{v_s, v_e\}$ donde se resuelven $|V| - 2$ problemas de corte mínimo. Para evitar confusiones, llamamos *nodo* y *conexión* a los vértices y arcos de D^k_{flow} y dejamos vértice y arco para los del dígrafo original D.

Sea $k \in V \setminus \{v_s, v_e\}$ un cliente. Suponemos que k está fijo en la parte restante de la demostración porque el algoritmo debe ejecutarse una vez para cada valor de k. Sea $(\bar{x}, \bar{y})$ una solución fraccionaria de CTTBF, F^* la desigualdad más violada (3.48). Finalmente, sea $\Delta_k(F) = \sum_{(i,k,m)\in F\cup\hat{N}^+(F)} x^m_{ik} - \sum_{(i,j,m)\in\hat{N}^+(F)} x^m_{ij}$ la violación de la desigualdad (3.48) definida por (F,k) donde $F \subseteq \{(i,j,m) \in \hat{A} : j \neq v_e\}$.

Definimos la red dirigida capacitada $D^k_{flow} = \langle V^k_{flow}, A^k_{flow}, s, t, c\rangle$ como la red con dos nodos $w^-_e \in V^k_{flow}$ y $w^+_e \in V^k_{flow}$, el nodo de entrada de e y el nodo de salida de e respectivamente, para cada arco $e \in \hat{A}$, una fuente $s \in V^k_{flow}$ y un sumidero $t \in V^k_{flow}$. Adicionalmente, para cada arco $e \in \hat{A}$ existe una conexión $w^-_e \to w^+_e$. También, hay una conexión $s \to w^-_{ikm}$ desde la fuente s a cada arco $ikm \in \hat{A}$ que llega a k, y otra conexión $w^+_{iv_em} \to t$ desde cada arco $iv_em \in \hat{A}$ que llega a v_e hacia el sumidero t.

$$V^k_{flow} = \{w^+_e : e \in \hat{A}\} \cup \{w^-_e : e \in \hat{A}\} \cup \{s, t\}, \tag{3.54}$$

$$\begin{aligned} A^k_{flow} = {} & \{s \to w^-_{ikm} : (i,k,m) \in \hat{A}\} \cup \{w^+_{iv_em} \to t : (i,v_e,m) \in \hat{A}\} \\ & \cup \{w^-_e \to w^+_e : e \in \hat{A}\} \cup \{w^+_e \to w^-_f : f \in \hat{N}^+(e), e \in \hat{A}\}. \end{aligned} \tag{3.55}$$

Asignamos la capacidad de las conexiones $w^-_e \to w^+_e$ como el flujo en $\bar{x}$ para el arco e, y como infinito para todas las demás conexiones.

$$c(v_1 \to v_2) = \begin{cases} \bar{x}_e & \text{si } v_1 = w^-_e, v_2 = w^+_e \text{ para algún } e \in \hat{A} \\ \infty & \text{en otro caso} \end{cases} \tag{3.56}$$

A continuación, demostramos que el corte de capacidad mínima S^* de D^k_{flow} está de alguna manera relacionado con la desigualdad más violada F^*. En particular, se cumple que $c(S^*) = \bar{y}_k - \Delta_k(F^*)$.

La demostración se divide en dos partes. Primero, exhibimos un corte con capacidad a lo sumo $\bar{y}_k - \Delta_k(F^*)$, lo que implica que $c(S^*) \leq \bar{y}_k - \Delta_k(F^*)$. Luego concluimos nuestra demostración mostrando que dado el corte mínimo S^* es posible construir un conjunto válido de arcos $F(S^*)$ tal que $\bar{y}_k - \Delta_k(F(S^*)) = c(S^*)$, lo que a su vez implica que $c(S^*) \geq \bar{y}_k - \Delta_k(F^*)$.

Primero, sea $S' = \{w^-_e \to w^+_e : e \in \hat{N}^+(F^*)\} \cup \{w^-_{ikm} \to w^+_{ikm} : (i,k,m) \notin F^* \cup \hat{N}^+(F^*)\}$ un conjunto de conexiones de D^k_{flow}. Mostraremos que la capacidad de S' es a lo sumo $\bar{y}_k - \Delta_k(F^*)$ y que contiene un corte de D^k_{flow}, que tiene a lo sumo esa capacidad. Por definición, la capacidad de S' es la suma de las capacidades de sus conexiones, por lo tanto su capacidad está dada por

$$\begin{aligned} c(S') &= \sum_{e\in\hat{N}^+(F^*)} c(w^-_e \to w^+_e) + \sum_{(i,k,m)\notin F^*\cup\hat{N}^+(F^*)} c(w^-_{ikm} \to w^+_{ikm}) \\ &= \sum_{e\in\hat{N}^+(F^*)} \bar{x}_e + \sum_{(i,k,m)\notin F^*\cup\hat{N}^+(F^*)} \bar{x}_{ikm} \\ &= \sum_{e\in\hat{N}^+(F^*)} \bar{x}_e + \bar{y}_k - \sum_{(i,k,m)\in F^*\cup\hat{N}^+(F^*)} \bar{x}_{ikm} \\ &= \bar{y}_k - \Delta_k(F^*). \end{aligned} \tag{3.57}$$

Hemos probado que $c(S') = \bar{y}_k - \Delta_k(F^*)$, ahora procedemos a mostrar que $D^k_{flow} \setminus S'$ desconecta s de t. Sea $s \to w^-_{e_1} \to w^+_{e_1} \to w^-_{e_2} \to w^+_{e_2} \to \cdots \to w^-_{e_r} \to w^+_{e_r} \to t$ un camino arbitrario de s a t. Sabemos que e_1 debe ser un arco de entrada (i,k,m) a k ya que está conectado a s, si $(i,k,m) \notin F^*$ entonces $w^-_{e_1} \to w^+_{e_1} \in S'$ por definición. Supongamos que $(i,k,m) \in F^*$, entonces también sabemos que e_r es un arco (i', v_e, m') que llega a v_e ya que está conectado a t. Además, sabemos que $(i', v_e, m') \notin F^*$ por definición de F^*. Entonces tenemos una secuencia de arcos $e_1, \cdots, e_r$ tal que $e_1 \in F^*$ y $e_r \notin F^*$ y cada uno es sucesor del anterior, luego algún arco e_l en el medio debe estar en $\hat{N}^+(F)$ y por lo tanto $w^-_{e_l} \to w^+_{e_l} \in S'$. Es fácil ver que S' debe contener un corte $s-t$ de D^k_{flow} con capacidad a lo sumo $\bar{y}_k - \Delta_k(F^*)$, entonces concluimos que $c(S^*) \leq \bar{y}_k - \Delta_k(F^*)$.

A continuación mostramos que $c(S^*) = \bar{y}_k - \Delta_k(F(S^*))$ donde $F(S^*) = \{(i,j,m) \in \hat{A} : w^+_{ijm} \in R\}$. Para ello, utilizamos el resultado de la ecuación (3.57) y demostramos que una conexión $w^-_e \to w^+_e \in S^*$ si y sólo si $e \in \hat{N}^+(F(S^*)) \cup \{(i,k,m) \notin F(S^*) \cup \hat{N}^+(F(S^*))\}$.

Primero, notar que $w^-_{ikm} \in R$ para todo $(i,k,m) \in \hat{A}$ y $w^+_{iv_em} \in T$ para todo $(i,v_e,m) \in \hat{A}$, de lo contrario al menos un arco infinito $s \to w^-_{ikm}$ o $w^+_{iv_em} \to t$ pertenecería a S^*. Esto implica que $F(S^*)$ no contiene ningún arco que llegue al depósito final y define una desigualdad válida (3.48).

- **Caso 1:** $e \in \{(i,k,m) \notin F(S^*)\} \Rightarrow w^-_e \to w^+_e \in S^*$. Dado que $e \notin F(S^*)$ entonces $w^+_e \in T$. Además, como e es un arco que entra al vértice k, tenemos que $w^-_e \in R$.

- **Caso 2:** $e \in \hat{N}^+(F(S^*)) \Rightarrow w^-_e \to w^+_e \in S^*$. Como $e \in \hat{N}^+(F(S^*))$ sabemos que $e \notin F(S^*)$ y, como resultado, $w^+_e \in T$. Además, debe existir un predecesor $f \in \hat{A}$ de e que pertenece a $F(S^*)$, lo que implica que $w^+_f \in R$. Dado que la conexión $w^+_f \to w^-_e$ tiene capacidad infinita, $w^-_e \in R$.

- **Caso 3:** $w^-_e \to w^+_e \in S^* \Rightarrow e \in \hat{N}^+(F(S^*)) \cup \{(i,k,m) \notin F(S^*)\}$. Como $w^+_e \in T$, sabemos que $e \notin F(S^*)$, por lo tanto, el caso cuando e entra al vértice k se cumple trivialmente. Cuando e no entra al vértice k, sabemos que dado que R está conectado, debe existir un predecesor $w^+_f \in V^k_{flow}$ de w^-_e que pertenece a R. Por lo tanto, $f \in F(S^*)$ y, dado que $e \notin F(S^*)$, tenemos que $e \in \hat{N}^+(F(S^*))$.

Dado que $c(S^*) \leq \bar{y}_k - \Delta_k(F^*)$ y $c(S^*) = \bar{y}_k - \Delta_k(F(S^*))$, concluimos que $F(S^*) = F^*$. Finalmente, la complejidad del algoritmo es la complejidad de resolver $|V| - 2$ problemas de corte mínimo en redes con $\mathcal{O}(|\hat{A}|)$ vértices y, por lo tanto, el peor caso tiene una complejidad de $\mathcal{O}(|V||\hat{A}|^3)$. □

3.A.2. Tablas detalladas de resultados

	CTTBF-basic				CTTBF-TDCESPP			
class	time	rLB	UB	nodes	time	rLB	UB	nodes
C101_25_0	74.03	-4760.59	-4587.17	16235	99.65	-4729.10	-4587.17	12469
C101_25_a	9.69	-2066.09	-2005.46	2250	6.21	-2005.48	-2005.46	0
C101_25_b	85.68	-1243.40	-1151.78	18210	26.93	-1209.63	-1151.78	4749
C101_25_c	122.56	-848.16	-747.57	20260	38.71	-828.19	-747.57	3517
C101_25_d	439.52	-155.22	-58.98	68090	72.31	-128.57	-58.98	6758
R101_25_0	178.12	-4577.54	-4046.79	15480	214.93	-4561.84	-4046.79	17394
R101_25_a	208.40	-2056.37	-1702.54	27854	313.21	-2024.76	-1702.54	29999
R101_25_b	188.40	-1335.89	-1026.83	38842	393.30	-1334.22	-1026.83	61453
R101_25_c	393.16	-278.66	-70.50	77525	396.24	-255.36	-70.50	70404
R101_25_d	467.77	-239.00	-67.43	94763	498.33	-231.13	-67.43	79266
RC101_25_0	1585.72	-7640.11	-6795.48	157788	1292.49	-7589.78	-6795.48	106380
RC101_25_a	181.48	-4019.28	-3560.45	13126	52.38	-3986.62	-3560.45	2138
RC101_25_b	90.38	-1559.38	-1349.78	9637	65.25	-1549.18	-1349.78	5217
RC101_25_c	66.37	-1937.91	-1797.49	4990	19.68	-1910.77	-1797.49	721
RC101_25_d	1017.89	-299.36	-117.58	91203	410.68	-280.52	-117.58	29394
C201_25_0	4209.85	-8497.98	-8311.96	1512987	2172.99	-8484.65	-8311.96	488389
C201_25_a	43.23	-4068.40	-3986.00	15429	25.65	-4062.55	-3986.00	6599
C201_25_b	1059.50	-1731.87	-1601.90	376054	176.29	-1711.96	-1601.90	29854
C201_25_c	4819.68	-606.49	-490.39	1284064	656.47	-580.81	-490.39	152538
C201_25_d	4410.04	-399.15	-263.87	1287698	899.73	-372.95	-263.87	182233
R201_25_0	-	-7675.10	-7444.62	1737260	1874.64	-7644.72	-7444.62	533994
R201_25_a	74.59	-2591.79	-2527.10	9212	31.76	-2574.57	-2527.10	2913
R201_25_b	66.70	-903.82	-804.15	7802	17.60	-825.75	-804.15	187
R201_25_c	49.79	-282.13	-220.00	7040	9.25	-220.00	-220.00	0
R201_25_d	21.87	-74.42	-8.43	12478	11.39	-32.10	-8.43	356
RC201_25_0	817.28	-13338.28	-13196.97	301606	241.76	-13310.98	-13196.97	71236
RC201_25_a	27.10	-1379.66	-1353.99	2449	10.22	-1357.63	-1353.99	5
RC201_25_b	71.68	-1082.35	-1031.59	8662	13.03	-1047.80	-1031.59	69
RC201_25_c	20.92	-879.38	-867.41	241	8.63	-867.41	-867.41	0
RC201_25_d	811.27	-451.04	-384.37	101092	32.88	-432.69	-384.37	3730
Totals	745.26			192519.55	283.03			47171.31

Tab. 3.A.1: Detailed results on TDCESPP instances.

	CTTBF-basic				CTTBF-TDOP			
class	time	rLB	UB	nodes	time	rLB	UB	nodes
1.a	11.46	-126.13	-115	1300	9.62	-126.13	-115	732
1.b	12.39	-149.06	-135	1921	13.90	-149.41	-135	2847
1.c	20.95	-172.66	-160	3920	6.59	-170.58	-160	125
1.d	7.15	-195.69	-185	1003	6.89	-195.90	-185	178
1.e	29.54	-220.33	-210	2716	25.39	-219.31	-210	4137
1.f	175.79	-248.36	-230	14939	283.33	-248.09	-230	15648
1.g	428.82	-266.27	-250	47509	434.62	-265.25	-250	31908
1.h	-	-282.63	-265	345841	1738.25	-281.88	-270	63655
1.i	-	-285.00	-255	134909	-	-285.00	-260	30560
2.a	2.36	-100.00	-100	0	4.25	-130.33	-100	696
2.b	3.31	-178.19	-150	472	3.65	-178.16	-150	123
2.c	2.37	-195.00	-195	0	2.52	-195.00	-195	0
2.d	5.09	-243.71	-220	114	3.89	-220.00	-220	0
2.e	6.10	-300.07	-260	1755	5.99	-299.93	-260	1714
2.f	5.33	-329.54	-310	142	3.62	-310.00	-310	0
2.g	8.89	-362.05	-340	1878	7.21	-354.72	-340	472
2.h	20.80	-418.42	-375	7150	24.10	-415.40	-375	5820
2.i	308.33	-450.00	-425	52493	274.88	-450.00	-425	20176
3.a	7.93	-417.61	-370	42	12.60	-412.35	-370	1271
3.b	17.29	-482.45	-420	3766	24.35	-485.17	-420	4555
3.c	15.50	-539.68	-500	1935	23.84	-539.68	-500	3963
3.d	13.62	-587.75	-560	1340	8.57	-587.75	-560	384
3.e	14.23	-634.31	-620	1147	24.49	-634.26	-620	992
3.f	35.90	-680.46	-650	2766	31.55	-679.27	-650	1307
3.g	72.98	-720.43	-690	6950	156.65	-718.46	-690	11807
3.h	-	-773.35	-720	208822	-	-772.72	-720	141583
3.i	1331.46	-799.64	-770	84681	-	-798.49	-770	124120
Totals	53.31			6750.35	60.54			4732.83

Tab. 3.A.2: Resultados detallados en instancias TDOP con restricción de makespan.

	CTTBF-basic				CTTBF-TDOP			
class	time	rLB	UB	nodes	time	rLB	UB	nodes
1.a	108.66	-136.47	-125.00	12381	116.65	-135.54	-125	7951
1.b	100.17	-162.29	-145.00	8346	183.43	-159.28	-145	11123
1.c	1119.91	-187.57	-165.00	158437	796.43	-184.86	-165	56669
1.d	257.03	-212.63	-190.00	25573	1214.27	-209.98	-190	90321
1.e	632.54	-233.61	-220.00	61519	348.06	-231.68	-220	20091
1.f	672.13	-258.72	-240.00	59116	442.57	-257.78	-240	28829
1.g	1456.44	-274.48	-260.00	123847	1123.83	-273.13	-260	42092
1.h	-	-285.00	-270.00	75080	-	-285.00	-275	63761
1.i	-	-285.00	-280.00	107402	-	-285.00	-275	83971
2.a	11.28	-179.06	-150.00	3118	12.01	-179.02	-150	3518
2.b	5.88	-220.38	-195.00	47	3.58	-195.00	-195	0
2.c	9.00	-261.68	-220.00	1811	8.17	-261.68	-220	1152
2.d	8.40	-302.98	-265.00	1191	8.32	-302.98	-265	1562
2.e	5.80	-341.25	-315.00	59	7.25	-344.28	-315	574
2.f	5.96	-385.58	-375.00	1143	7.78	-385.01	-375	533
2.g	2.20	-415.00	-415.00	0	3.10	-415.00	-415	0
2.h	41.77	-450.00	-440.00	6261	38.10	-450.00	-440	3959
2.i	25.50	-450.00	-450.00	3990	16.93	-450.00	-450	2871
3.a	95.56	-437.11	-430.00	7986	65.40	-437.11	-430	3856
3.b	44.84	-504.38	-490.00	3296	33.48	-504.38	-490	1561
3.c	38.29	-553.90	-550.00	2901	52.52	-553.90	-550	2276
3.d	66.04	-600.19	-590.00	4076	49.04	-600.19	-590	1894
3.e	60.20	-646.71	-630.00	4387	110.03	-646.71	-630	5164
3.f	83.04	-692.32	-680.00	5036	88.61	-692.32	-680	4489
3.g	83.45	-738.37	-730.00	5166	55.69	-738.37	-730	2335
3.h	84.19	-778.40	-770.00	5083	89.75	-778.39	-770	4901
3.i	217.55	-800.00	-800.00	7056	427.32	-800.00	-800	11890
Totals	209.43			20473.04	212.09			12384.44

Tab. 3.A.3: Resultados detallados en instancias TDOP con restricción de duración.

4. PROGRAMACIÓN DINÁMICA PARA EL PROBLEMA DEL VIAJANTE DE COMERCIO CON VENTANAS DE TIEMPO Y DEPENDENCIA TEMPORAL

4.1. Introducción

La creciente prevalencia de las entregas directas al consumidor ha resaltado el papel crucial de la logística de última milla en la planificación urbana. En este capítulo, revisamos los problemas TDTSP y TDTSPTW, enfocándonos en un enfoque algorítmico diferente.

Recordemos que el TDTSPTW se define de la siguiente manera. Sea $D = (V, A)$ un dígrafo con un conjunto de vértices $V = \{0, 1, \ldots, n+1\}$, donde 0 y $n+1$ representan el depósito inicial y final, respectivamente, y $V\backslash\{0, n+1\}$ representa a los clientes. Las operaciones se llevan a cabo dentro de un *horizonte de planificación* $[0, T]$. Asociado a cada $v \in V$, existe un intervalo de tiempo $[a_v, b_v] \subseteq [0, T]$ que especifica la *ventana de tiempo* en la que v debe ser *visitado*; a_v y b_v son el *tiempo de apertura* y el *tiempo de cierre* de v, respectivamente. Cabe destacar que v debe ser *alcanzado* antes de ser visitado. Si se alcanza antes de a_v, entonces la visita debe ser demorada hasta un tiempo $t \in [a_v, b_v]$.

Revisamos brevemente algunas definiciones de la Sección 2.5 que se utilizan a lo largo de este capítulo para mayor comodidad del lector. Consideramos el modelo de tiempo de viaje propuesto por Ichoua et al. [16], donde se define una *función de tiempo de viaje* τ_{vw} con dominio $[0, T]$ para cada arco $(v, w) \in A$. Para $t \in [0, T]$, $\tau_{vw}(t)$ es el tiempo requerido para viajar de v a w cuando se parte de v en el tiempo t. Cada τ_{vw} es continua, lineal a trozos, y satisface la *propiedad first-in first-out* (FIFO), es decir, $t + \tau_{vw}(t) \leq t' + \tau_{vw}(t')$ para todo $0 \leq t \leq t' \leq T$. Además, las funciones de tiempo de viaje también satisfacen la *propiedad de no espera* definida en la Sección 2.5.3, lo que significa que $t + \tau_{vw}(t) \geq a_w$ para $t \in [0, T]$. En otras palabras, ningún vértice puede ser alcanzado antes de su tiempo de apertura. Sin embargo, la familia de funciones de tiempo de viaje no necesita satisfacer la *desigualdad triangular*; por lo tanto, $\tau_{vz}(t)$ podría ser mayor que $\tau_{vw}(t) + \tau_{wz}(t + \tau_{vw}(t))$ para algunos v, w, z.

Sea $p = (v_0, \ldots, v_k)$ un camino en D. Definimos δ_p como la *función de tiempo de finalización* de p, que es continua, lineal a trozos y no decreciente. El valor $\delta_p(t)$ denota el tiempo más temprano en el que el camino p puede completarse al partir de v_0 en el tiempo t. También definimos $\mathrm{FDT}(p)$ como el conjunto de *tiempos de partida factibles*, es decir, el conjunto de todos los tiempos t tales que partir de v_0 en el tiempo t resulta en un recorrido factible del camino. La duración de p al partir de v_0 en el tiempo t está dada por $\Gamma_p(t) = \delta_p(t) - t$, que representa el tiempo mínimo requerido para recorrer el camino comenzando en el tiempo t. Dado que tanto δ_p como la función identidad $t \mapsto t$ son continuas y función lineal a trozos, se sigue que Γ_p también es continua y lineal a trozos.

Decimos que un camino p es: *factible* si tiene un tiempo de partida factible; un *tour* si comienza en 0, termina en $n+1$, y visita exactamente $n+2$ vértices; y *elemental* si es factible y ningún vértice es recorrido dos veces. Cuando p es factible, su *makespan* $\delta_p(a_{v_0})$ es el tiempo más temprano en el que se puede alcanzar v_k, mientras que su *duración* es el tiempo mínimo requerido para viajar de v_0 a v_k respetando todas las ventanas de tiempo. Es decir, la duración es $c_p = \min\{\Gamma_p(t) \mid t \in \mathrm{FDT}(p)\}$.

La duración definida anteriormente corresponde a un recorrido *forward* de un camino. Sin embargo, en este capítulo también requerimos la capacidad de evaluar el costo de un camino cuando se conoce el tiempo de llegada a su último vértice. Para mantener la consistencia, introducimos la noción de la *duración backward* de un camino. Como se recordó en la Sección 2.5.2, las definiciones backward reflejan sus contrapartes forward.

Definimos $\mathrm{FAT}(p)$ como el conjunto de *tiempos de llegada factibles* para la ruta p, lo que significa que $t \in \mathrm{FAT}(p)$ si existe un recorrido factible de p que termina en el tiempo t. De manera análoga a la función de tiempo de llegada δ_p, definimos la *función de tiempo de partida* $\gamma_p : \mathrm{FAT}(p) \to \mathrm{FDT}(p)$, donde $\gamma_p(t)$ denota el tiempo más tardío en el que se puede partir desde v_0 y aún así completar p

para el tiempo t. Siendo el contraparte de δ_p, no es difícil ver que γ_p también es continua, lineal a trozos y no decreciente. Notar que, por definición, $\Delta_p(t) = t - \gamma_p(t)$ es el tiempo mínimo requerido para recorrer p cuando v_k es visitado en el tiempo $t \in \mathrm{FAT}(p)$ (Figura 2.4d). El dominio de Δ_p es $\mathrm{FAT}(p)$ y, por uniformidad, nos referimos a Δ_p como la *duración de p en v_k*.

Por definición, p es factible si y sólo si $\mathrm{FAT}(p) \neq \varnothing$ y, en tal caso, su makespan es $\delta_p(a_{v_0}) = \min \mathrm{FAT}(p)$. Además, no es difícil ver que la duración c_p de p corresponde al mínimo entre las duraciones $\Delta_p(t)$ en v_k, es decir, $c_p = \min\{\Delta_p(t) \mid t \in \mathrm{FAT}(p)\}$.

El objetivo en el TDTSPTW es encontrar un tour elemental de duración mínima. El TDTSPTW$_m$, en el que se minimiza el makespan, se resuelve dejando $b_0 = a_0$; el TDTSP y el TDTSP$_m$ se obtienen eliminando las ventanas de tiempo; y el *Problema de Duración Mínima de Tour* (MTDP, por sus siglas en inglés) se resuelve cuando las funciones de tiempo de viaje son constantes (see Tilk et al. [27]).

Las principales contribuciones de este capítulo son las siguientes. Desarrollamos nuevos algoritmos de programación dinámica para el TDTSPTW, basándonos en los trabajos de Baldacci et al. [9] para el TSPTW y de Tilk et al. [27] para el MTDP. En el caso del MTDP, Tilk et al. [27] utilizan tres números para describir el dominio completo de la función de duración de cada camino. Motivados por el contexto con dependencia temporal, en nuestro enfoque describimos la duración de cada camino mediante una función parcial, cuyo dominio se restringe a los tiempos de llegada no dominados. Esta representación —que no había sido considerada previamente para variantes del TDTSP— permite aplicar una regla de dominancia parcial que reduce considerablemente la cantidad de estados enumerados.

Además, introducimos una nueva relajación del espacio de estados diseñada específicamente para problemas TDTSP, una nueva regla de preprocesamiento, nuevas reglas de factibilidad dependientes del tiempo para descartar etiquetas, y un esquema alternativo de procesamiento de etiquetas que resulta menos sensible a la calidad de las cotas.

Llevamos a cabo experimentos computacionales extensivos sobre instancias con ventanas de tiempo estrictas, amplias o inexistentes, propuestas para el TDTSPTW, el TDTSP y el MTDP. Los resultados muestran que nuestro algoritmo de programación dinámica supera a los mejores algoritmos exactos previamente propuestos en la literatura para cada caso, encontrando una cantidad significativa de nuevas soluciones óptimas. En general, el algoritmo logra manejar eficazmente instancias con alto grado de flexibilidad respecto al tamaño de las ventanas de tiempo, lo que sugiere una nueva dirección de investigación para el TDTSP y sus variantes.

El resto del capítulo está organizado de la siguiente manera. La Sección 4.3 ofrece una visión general de algunos conceptos clave respecto a las definiciones de labels. El algoritmo de labeling para el TDTSPTW se presenta en la Sección 4.4. El paso de preprocesamiento aplicado para reducir el tamaño de la instancia y para calcular la información requerida para el acotamiento se expone en la Sección 4.5. El acotamiento se describe en la Sección 4.6, los algoritmos desarrollados se discuten en la Sección 4.7, y los experimentos numéricos y sus resultados se reportan en la Sección 4.8. Finalmente, las conclusiones y algunas líneas de investigación futura se presentan en la Sección 4.9.

4.2. Revisión de la literatura

El TSP y el TSPTW han sido ampliamente estudiados en la literatura junto con otras variantes menos populares, como el MTDP. Los algoritmos exactos tradicionales se basan en técnicas poliedrales y algoritmos BC (e.g., Applegate et al. [2] y Ascheuer et al. [6, 7]). Más recientemente, [12] propusieron la *formulación de intervalos de tiempo* para obtener cotas más ajustadas y derivar nuevas familias de desigualdades válidas.

La Programación Dinámica también se ha implementado con éxito para el TSPTW y el MTDP. [10] introdujo las llamadas *relajaciones en el espacio de estados* para calcular cotas inferiores rápidas para el TSPTW y otros problemas de ruteo. [25] diseñó un algoritmo para el MTDP y realizó una comparación con el TSPTW y otras variantes. Más adelante, [13] propuso el primer algoritmo de DP para el TSPTW, incluyendo un análisis exhaustivo para reducir el número de

estados enumerados. [20] propuso un algoritmo diferente de DP para el TSPTW con Restricciones de Precedencia (TSPTW-PC) que se basa en la idea de [10] y utiliza cotas inferiores para reducir el número de estados explorados; esta idea es conocida como *bounding*.

La relajación del espacio de estados *ng-path*, introducida por [8] para el *VRP con Ventanas de Tiempo* (VRPTW, por sus siglas en inglés), se ha convertido en un estándar para problemas de ruteo. [9] adaptó la relajación *ng*-path al TSPTW, introduciendo las relajaciones *ng-tour* y *ngP-tour*. Su marco incluye un *método de penalización*, basado en una relajación Lagrangiana, que mejora las cotas inferiores y reduce el grafo del espacio de estados. [27] adaptó este esquema al MTDP, introduciendo relajaciones adicionales a las relajaciones *ng*-tour y *ngP*-tour. Además, mejoran el marco general aplicando una técnica llamada *Dynamic Neighbour Augmentation* (DNA) para obtener cotas inferiores similares en menos tiempo de cómputo.

Ha habido varios intentos de capturar la congestión utilizando diferentes modelos; véase por ejemplo [15]. El *modelo de velocidad de viaje* Ichoua et al. [16] ha sido ampliamente adoptado por la comunidad de optimización de ruteo. Este modelo asume que las operaciones se realizan en un horizonte de planificación que se divide en zonas horarias donde la velocidad promedio de viaje de cada arco permanece constante. La función de tiempo de viaje de cada arco se obtiene entonces procesando las velocidades de las diferentes zonas horarias. Como se discute en la Sección 4.1, cada función de tiempo de viaje es continua, lineal a trozos, y satisface la propiedad FIFO.

La investigación sobre el TDTSP$_m$ fue inicialmente motivada por un algoritmo BC propuesto por [11], que luego fue ampliado por [3] y [1] con nuevos métodos para calcular cotas inferiores más ajustadas. Se observa una secuencia similar para el TDTSPTW$_m$, donde el enfoque de [11] fue adaptado por [4] y [5]. Estos últimos enfoques dependen en gran medida de la calidad de las cotas inferiores, las cuales se demuestra que son ajustadas bajo ciertas condiciones de tráfico. [21] se basan en un MILP propuesto por [26] y desarrollan un BC para el TDTSPTW$_m$, obteniendo resultados competitivos con el estado del arte. [18, 17] extienden el enfoque de [21] para el TDTSPTW, TDTSPTW$_m$ y otras variantes. Su formulación reduce el tamaño del MILP mientras preserva la calidad de la relajación LP, y se fortalece con nuevas familias de desigualdades válidas. Alternativamente, [28] proponen un algoritmo de *Dynamic Discretization Discovery* (DDD) para el TDTSPTW y el TDTSPTW$_m$.

Debido a la publicación simultánea de varios artículos, no existía en ese momento un benchmark integral que evaluara todos los métodos disponibles a lo largo del conjunto completo de instancias. En cuanto al TDTSP$_m$, los mejores resultados en el momento de la publicación del artículo asociado con este capítulo fueron proporcionados por [1]. Los algoritmos propuestos por [28] y [17] representaban el estado del arte para métodos exactos que abordan el TDTSPTW$_m$ y el TDTSPTW, respectivamente, mientras que [5] también debía ser considerado en el contexto del TDTSPTW$_m$. Los resultados sugieren que los métodos de [28] y [17] son más robustos en instancias con ventanas de tiempo estrictas, mientras que el de [3] funciona mejor cuando las ventanas de tiempo son más amplias. Hasta donde sabemos, en el momento de la publicación, no se había propuesto ningún enfoque basado en programación dinámica para el TDTSPTW o problemas relacionados.

4.3. Labels parciales y el criterio de dominancia parcial: una visión general

Esta sección resume las discusiones sobre la implementación de la dependencia temporal en algoritmos de labeling, tal como se presentó inicialmente en Lera-Romero et al. [19]. Para mayor completitud y facilidad de lectura, incluimos aquí también las partes relevantes. Como se describe en la Sección 2.7, los algoritmos de labeling forward para problemas independientes del tiempo usualmente funcionan manteniendo un único label para cada camino elemental que comienza en el depósito 0. Computacionalmente, un label es una estructura que describe un camino y el costo requerido para recorrerlo. Aquí, el costo es una constante que es independiente del momento en que se visita un vértice. Partiendo de un label inicial que representa el camino (0), el algoritmo extiende iterativamente cada label enumerado para considerar todos los caminos elementales que visitan un vértice adicional. Para evitar recorrer todos los caminos, se aplican reglas eficientes para

(a) Reducción *(b)* Fragmentación

Fig. 4.3.1: Si se mantienen los labels totales y se aplica un criterio de dominancia total, entonces los labels en (a) y (b) permanecen sin cambios. Si se aplica un criterio de dominancia parcial con labels totales, entonces los labels negros en (a) y (b) pueden descartarse, pero el label verde se mantiene intacto. Finalmente, si se mantienen labels parciales, entonces las porciones punteadas en (a) y (b) se eliminan. En este caso, el dominio del label verde se reduce. Esto usualmente implica que se necesitan mantener menos piezas de la función de duración (6 vs. 1 en a), aunque a veces se requieren más piezas (1 vs. 2 en b).

descartar labels acotados y dominados. Considere un label L que representa un camino p_L, y sea p un camino parcial factible que termina en el vértice v_{n+1} tal que el camino $p_L + p$ es un tour elemental. Decimos que L está *acotado* cuando el costo de todo tour elemental $p_L + p$ es mayor que una cota superior dada *ub*, mientras que está *dominado* por un label M, que representa un camino p_M, cuando $p_M + p$ es un tour elemental con un costo no mayor que el costo de $p_L + p$.

El algoritmo de labeling anterior puede generalizarse de diferentes maneras al contexto con dependencia temporal. Una forma sencilla es considerar que cada camino p está representado por un *label total* L que, como antes, es una estructura que codifica p y su costo de recorrido. Esta vez, sin embargo, el costo es una función definida para **cada** tiempo de llegada factible en $\mathrm{FAT}(p)$. Tanto en el MTDP como en el TDTSPTW, la función de costo es Δ_p, es decir, la duración de p en su último vértice v (Figura 2.4d). Aunque computacionalmente L es una estructura, a veces es mejor pensar en L como una función, con dominio $\mathrm{FAT}(p)$, tal que $L(t)$ es el label "independiente del tiempo" que representa p cuando v es visitado en el tiempo t. Esta visión *conceptual* nos permite generalizar fácilmente diferentes nociones del contexto independiente del tiempo. Por ejemplo, si $L(t)$ está dominado por $M(t)$ para algún label total M, entonces ningún tour elemental que recorra p y visite v en el tiempo t tiene una duración menor que algún tour elemental obtenido al extender el camino de M. Esto implica que $L(t)$ puede descartarse de forma segura, lo que nos lleva a la siguiente generalización de la relación de dominancia independiente del tiempo: L está *totalmente dominado* por un label M, que representa un camino q, cuando $\mathrm{FAT}(p) \subseteq \mathrm{FAT}(q)$ y $L(t)$ está dominado por $M(t)$ para todo $t \in \mathrm{FAT}(p)$ (Figura 4.3.1). En un algoritmo de labeling con un *criterio de dominancia total*, un label L se descarta sólo si está totalmente dominado por otro label total M.

Dado que $L(t)$ no puede generar una ruta óptima cuando $L(t)$ está dominado por $M(t)$, se sigue que L puede ser descartado si existe un conjunto de labels $\mathcal{M}$ tal que, para todo $t \in \mathrm{FAT}(p)$, $L(t)$ está dominado por $M(t)$ para algún $M \in \mathcal{M}$ (Figura 4.3.1). Sin embargo, L no se descarta cuando se aplica el criterio de dominancia completo, a menos que uno de los labels en $\mathcal{M}$ domine completamente a L. Una segunda alternativa para un algoritmo de labeling con dependencia temporal es aplicar un *criterio de dominancia parcial*. Bajo este criterio, un label total L puede ser descartado cuando, para todo $t \in \mathrm{FAT}(p)$, existe un label total M tal que $M(t)$ domina a $L(t)$.

La dominancia parcial es superior a la dominancia total, ya que se descartan más labels y, por lo tanto, se enumeran menos labels. Sin embargo, el uso de labels totales implica que $\Delta_p(t)$ debe mantenerse para cada $t \in \mathrm{FAT}(p)$, incluso cuando $L(t)$ está dominado por otro label. En consecuencia, muchas porciones dominadas de Δ_{p+w} $(w \in V)$ deben ser calculadas cuando L se extiende al label que representa la ruta $p + w$ (Figura 4.3.1a). Esta operación, sobre datos que se sabe que son subóptimos, puede consumir una gran cantidad de tiempo y memoria. El consumo de memoria es de suma importancia, teniendo en cuenta que los algoritmos de labeling requieren una cantidad exponencial de espacio en el peor caso. Una tercera alternativa para un algoritmo

de labeling con dependencia temporal, por lo tanto, es mantener *labels parciales*. Como antes, un label parcial es una estructura L que codifica una ruta p junto con su función de costo. Esta vez, sin embargo, la función de costo Δ_L se obtiene restringiendo Δ_p a un dominio no vacío $\mathrm{dom}(L) \subseteq \mathrm{FAT}(p)$ (Figura 4.3.1). En nuestra visión conceptual, L puede considerarse como una función parcial sobre el dominio $\mathrm{FAT}(p)$, mientras que el dominio de L es una familia de intervalos disjuntos. Aplicando un criterio de dominancia parcial, un tiempo de llegada factible $t \in \mathrm{dom}(L)$ puede descartarse de L cuando $L(t)$ está dominado, sin importar si $L(t')$ está dominado para $t' \in \mathrm{dom}(L) \setminus \{t\}$ (Figura 4.3.1).

Se afirma anteriormente que los labels parciales son preferidos sobre los labels totales porque se almacenan y procesan menos fragmentos de Δ_p a medida que se eliminan porciones mayores de Δ_L (Figura 4.3.1a). Sin embargo, en ciertos puntos del algoritmo, un único fragmento de Δ_p puede dividirse en muchos fragmentos disjuntos de Δ_L (Figura 4.3.1b). En consecuencia, este label parcial en particular requiere más espacio que su correspondiente label total y, por lo tanto, se consume más tiempo para procesarla. A pesar de estos casos particulares, desde una perspectiva global se almacenan menos fragmentos con labels parciales. Por ello, como se observa en los experimentos descritos en Lera-Romero et al. [19], la efectividad de los labels parciales supera ampliamente la explosión de labels/fragmentos que resulta de los labels totales y/o del criterio de dominancia total.

Una propiedad clave de los algoritmos independientes del tiempo es que cada label L está determinado de manera única por su camino representado p. Esta propiedad también se cumple en aquellos algoritmos con dependencia temporal que mantienen labels totales, ya que tanto $\mathrm{FAT}(p)$ como Δ_p están determinados de manera única por p. Esto no ocurre cuando se mantienen labels parciales, ya que existen infinitas posibilidades en las que $\mathrm{dom}(L)$ puede reducirse como resultado de una dominación. El hecho de que L esté desacoplado de p dificulta la implementación y descripción de los algoritmos que mantienen labels parciales, ya que se debe tener en cuenta que los labels observan sólo una porción del costo de su camino.

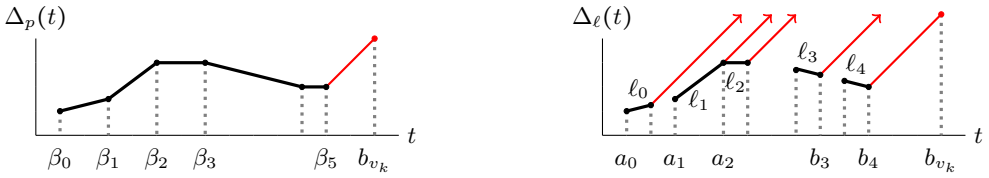
Aunque los labels parciales se describen como estructuras, esto es una simplificación excesiva: los labels parciales son un concepto que puede implementarse de muchas maneras diferentes. Lera-Romero et al. [19] examina tres formas de implementar labels parciales que son equivalentes en el sentido de que se descartan los mismos puntos de la función de duración por dominancia. Sin embargo, se emplean diferentes estructuras de datos, cada una con sus propias fortalezas y debilidades; ninguna de las implementaciones es estrictamente mejor que las otras para cada instancia posible. Los algoritmos que desarrollamos para el TDTSPTW se basan en los llamados labels lineales. Un label lineal ℓ es una tupla que codifica p cuya función de costo Δ_ℓ es lineal y describe sólo una de las piezas de la duración Δ_L de un label parcial L. Así, L es una secuencia de labels lineales $\ell_1, \ldots, \ell_k$ cuyas duraciones tienen dominios disjuntos. Cabe destacar que los labels parciales son sólo un concepto en nuestro algoritmo, ya que L no se almacena explícitamente y no existe una manera eficiente de reconocer si dos labels lineales corresponden al mismo camino. La ventaja de esta representación es que podemos mezclar labels lineales de diferentes caminos en un solo bucket para procesar labels de manera más eficiente. Además, los labels lineales proporcionan una forma simple y conveniente de implementar bounding. Este aspecto quedará más claro en las secciones siguientes con la descripción del algoritmo.

Nos referimos a Lera-Romero et al. [19] para una discusión más profunda sobre el labeling parcial y cómo se compara con el labeling total en una implementación real de un problema de camino mínimo con recursos restringidos.

4.4. Algoritmo exacto de programación dinámica

En esta sección desarrollamos el algoritmo principal que sirve como marco para resolver el TDTSPTW. El algoritmo tiene tres configuraciones según si se realiza una búsqueda Forward, Backward o bidireccional. Sólo se discute la búsqueda Forward; las adaptaciones para los escenarios Backward y bidireccional aparecen en los Apéndices 4.A.1 y 4.A.2.

Sea $p = (v_0, \cdots, v_k)$ un camino elemental extendido desde $v_0 = 0$. Recuerde que $\delta_p(t)$ es el tiempo más temprano en el que se puede visitar v_k cuando se parte de v_0 en el tiempo de partida factible



(a) Duración de p *(b)* Duración de labels para p

Fig. 4.4.1: (a) función de duración de una ruta $p = (0, v_1, \ldots, v_k)$ con puntos de quiebre $\beta_0, \ldots, \beta_6$ y un segmento de espera en rojo; por definición, $\beta_0 = \delta_p(\text{mín FDT}(p)) = \text{mín FAT}(p)$, $\beta_5 = \delta_p(\text{máx FDT}(p))$, y $\beta_6 = b_{v_k}$. (b) función de duración de cinco labels $\ell_0, \ldots, \ell_4$ para p, en algún paso del algoritmo. Aquí, $\text{dom}(\ell_i) = [a_i, b_i]$, cada línea negra representa la duración dentro de dom, y cada línea roja es el segmento de espera desde b_i hasta b_{v_k}.

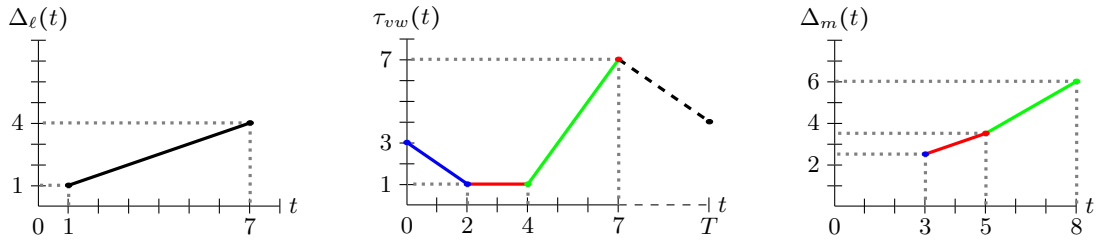
$t \in \text{FDT}(p)$. La función δ_p es continua y lineal a trozos, al igual que la función de duración Δ_p en v_k. Toda función continua y lineal a trozos f se caracteriza por una secuencia mínima de *puntos de quiebre* $\beta_0 < \ldots < \beta_j$ tal que la *pieza* obtenida al restringir el dominio de f al intervalo $[\beta_i, \beta_{i+1}]$ es lineal para $0 \le i < j$. La Figura 4.4.1a muestra Δ_p para el caso en que v_k siempre se alcanza antes de su tiempo de cierre. La pieza roja con pendiente 1, llamada la *pieza de espera*, representa el hecho de que la visita puede demorarse hasta cualquier momento antes de b_{v_k}. Este enfoque mejora la poda por dominancia, ya que un label puede dominar a otro solo en un tiempo específico al esperar en el último vértice.

Una *label forward lineal* de p es una tupla $\ell = (\text{prev}, v_k, S, \Delta)$ donde prev es un puntero a un label forward lineal de $(v_0, \ldots, v_{k-1})$, $S = \{v_0, \ldots, v_k\}$ es el conjunto de vértices visitados, y Δ es una función lineal en un dominio no vacío $\text{dom}(\ell)$ tal que $\Delta(t) = \Delta_p(t)$ para todo $t \in \text{dom}(\ell)$ (Figura 4.4.1b). Usamos $p(\ell)$, $\text{prev}(\ell)$, $v(\ell)$, $S(\ell)$, y Δ_ℓ para denotar p, prev, v_k, S, y Δ, respectivamente, mientras que omitimos los términos forward y lineal cuando no hay posibilidad de confusión. Por definición, $\text{dom}(\ell)$ es un intervalo $[a(\ell), b(\ell)]$; intuitivamente, $a(\ell)$ y $b(\ell)$ son los tiempos más tempranos y más tardíos en los que se puede alcanzar v_k, en la medida en que el conocimiento de ℓ se extiende. Como se discute en la Sección 4.3, ℓ representa solo una porción de Δ_p, por lo que se mantienen muchas labels para cada ruta.

Para cada camino elemental p, el algoritmo de labeling implícitamente mantiene una secuencia $L = \ell_0, \ldots, \ell_j$ de labels de p tal que $b(\ell_i) \le a(\ell_{i+1})$ para todo $1 \le i < j$ (Figura 4.4.1b). Además, o bien ℓ_i y ℓ_{i+1} corresponden a diferentes tramos de Δ_p o bien $b(\ell_i) < a(\ell_{i+1})$. Por lo tanto, $t \in \text{FAT}(p)$ pertenece como máximo a dos labels en L y t pertenece a dos labels sólo si t es un punto de quiebre de Δ_p. La secuencia L corresponde al label parcial de p. Reiteramos que L es sólo un concepto y no es mantenida por el algoritmo. Además, no es posible decidir en tiempo $O(1)$ si dos labels representan el mismo camino. Como se discute en la Sección 4.3, L y ℓ pueden considerarse como funciones. En esta interpretación, $\text{dom}(\ell)$ es el dominio de ℓ, $\text{dom}(L) = \bigcup_i \text{dom}(\ell_i) \subseteq \text{FAT}(p)$ es el dominio de L, y $L(t) = \ell(t)$ es un *label independiente del tiempo* $(\text{prev}(\ell), v(\ell), S(\ell), t, \Delta_\ell(t))$ para $t \in \text{dom}(\ell)$. La Figura 4.4.1b muestra la duración de cinco labels que representan un camino elemental p. El algoritmo de labeling nunca mantiene un label ℓ cuyo dominio esté incluido en el tramo de espera de p. En cambio, dado que la espera después de $b(\ell)$ siempre es posible en lo que respecta a ℓ, se asume que ℓ puede continuar con un *tramo de espera* de pendiente 1, como se muestra con las líneas rojas. Es decir, desde la perspectiva de ℓ, el conjunto $\text{FAT}(\ell) = [a(\ell), b_{v(\ell)}]$ de *tiempos de llegada factibles* para ℓ siempre contiene a $b_{v(\ell)}$, mientras que $\Delta_\ell(t) = \Delta_\ell(b(\ell)) + (t - b(\ell))$ es la duración de ℓ cuando $v(\ell)$ se alcanza en $t > b(\ell)$.

4.4.1. Extensión de labels

Comenzando con el *label inicial* $\ell_0 = (\perp, 0, \{0\}, \Delta_{(0)})$ que representa el camino (0), donde $\Delta_{(0)}(t) = 0$ para $t \in [a_0, b_0]$, el algoritmo *extiende* iterativamente cada label ℓ a través de cada arco $(v(\ell), w) \in A$ que corresponde a un vértice no visitado $w \notin S(\ell)$. El resultado de la extensión de ℓ



(a) Duración del label ℓ. *(b)* Tiempo de viaje de (v, w). *(c)* Extensiones de ℓ.

Fig. 4.4.2: (a) $\Delta_\ell(t)$ para un label ℓ con $v(\ell) = v$. (b) Función de tiempo de viaje τ_{vw} del arco (v, w) para un vértice $w \notin S(v)$ con ventana de tiempo $[2, 8]$; las piezas azul, roja y verde son las únicas que tienen una intersección no vacía con $\text{dom}(\ell)$. (c) Extensión de ℓ a través de (v, w) en la que se obtienen tres labels m_1 (azul), m_2 (rojo) y m_3 (verde). Nótese que Δ_{m_1} es un solo punto que se superpone con Δ_{m_2}. Esto ocurre porque si se parte de v en $t \le 2$, entonces se llega a w en el tiempo 3. Por la propiedad FIFO, la duración cuando w es visitado en el tiempo 3 se minimiza cuando v es visitado en el tiempo 2. Por lo tanto, $\Delta_{m_1}(t) = \Delta_\ell(2) + \tau_{v(\ell)w}(2) = 2{,}5$. Respecto a m_2, observar que $\text{dom}(m_2) = [3, 5]$, ya que estos son los tiempos de llegada factibles a w cuando se restringe a la segunda pieza de τ_{vw}. Para referencia, $\Delta_{m_2}(5) = 3{,}5 = \Delta_\ell(4) + \tau_{vw}(4)$.

a través del arco $(v(\ell), w)$ es una secuencia M de labels de $p(\ell) + w$. En términos coloquiales, las duraciones de los labels de M, tomadas en conjunto, definen una función Δ_M que representa la duración de $p(\ell) + w$ cuando $v(\ell)$ es visitado en un tiempo t, para cada $t \in \text{dom}(\ell)$ (Figura 4.4.2). Por supuesto, el dominio de M está restringido a aquellos $t \in \text{dom}(\ell)$ que respetan la ventana de tiempo $[a_w, b_w]$ de w.

Para formalizar la idea anterior, recuerde que $\tau_{vw}(t)$ representa el tiempo requerido para viajar desde el vértice v hasta el vértice w al partir en el tiempo $t \in [0, T]$ (Figura 4.4.2b). Debido a la propiedad de no espera, si el vértice v es visitado en el tiempo t, entonces w puede ser visitado no antes de $t + \tau_{vw}(t)$, y la duración total de la ruta se convierte en $\Delta_\ell(t) + \tau_{vw}(t)$.

Definimos la función Δ_M sobre el dominio $[a(\ell) + \tau_{vw}(a(\ell)),\ b(\ell) + \tau_{vw}(b(\ell))] \cap [a_w,\ b_w]$, tal que $\Delta_M(t + \tau_{vw}(t)) = \Delta_\ell(t) + \tau_{vw}(t)$ (Figura 4.4.2c). Intuitivamente, esta construcción propaga el label ℓ a través del arco (v, w) desplazando el tiempo de inicio y sumando el tiempo de viaje correspondiente.

Es importante destacar que Δ_M es continua y lineal a trozos, con una pieza por cada segmento de τ_{vw} que intersecta $\text{dom}(\ell)$. Cada label resultante $m \in M$ corresponde a una de estas piezas de Δ_M. Es decir, para la implementación, se genera un label $m = (\ell, w, S \cup \{w\}, \Delta_m)$ para cada pieza τ de τ_{vw} con dominio $[a(\tau), b(\tau)]$ tal que $[a(\tau), b(\tau)] \cap \text{dom}(\ell)$ es un intervalo no vacío $[a, b]$, y $\delta_\ell(a) + \tau(a) \le b_w$.

El dominio de Δ_m es $[a + \tau(a),\ \text{mín}\{b + \tau(b),\ b_w\}]$, y para cada t en este dominio, $\Delta_m(t) = \Delta_\ell(t') + \tau(t')$, donde $t' \in [a, b]$ es el mayor valor tal que $t' + \tau(t') \le t$.

4.4.2. Eliminación de labels

La eficiencia de los algoritmos de labeling depende de la capacidad de descartar la mayor cantidad de labels posible y lo más rápido posible. En esta sección generalizamos tres reglas altamente efectivas del contexto independiente del tiempo.

Se dice que un label ℓ independiente del tiempo es *tour-infactible* cuando su camino no puede extenderse a un tour elemental factible. Claramente, si ℓ es tour-infactible, entonces ℓ es inútil y puede descartarse. Como calcular todas las extensiones de $p(\ell)$ es tan costoso como ejecutar el algoritmo de labeling, se aplican reglas eficientes para detectar labels tour-infactibles. En este trabajo, se declara que ℓ es tour-infactible cuando algún vértice no visitado w no puede alcanzarse dentro de su ventana de tiempo, incluso si se toma el camino más rápido desde $v(\ell)$ hasta w. Como señala Baldacci et al. [9], esta regla es altamente efectiva para el TSPTW, especialmente en presencia de ventanas de tiempo estrictas.

La regla anterior se generaliza al contexto con dependencia temporal recordando que los labels codifican funciones: se elimina t de $\mathrm{dom}(\ell)$ cuando $\ell(t)$ es tour-infactible. En última instancia, se descarta ℓ si $\mathrm{dom}(\ell)$ queda vacío. La Regla 4.1 implementa esta idea sin recurrir a labels independientes del tiempo. Aquí, $\mathrm{LDT}(v, w)$ es el último tiempo en que se puede partir del vértice v para alcanzar un vértice w antes de su tiempo de cierre a través de cualquier camino.

Regla 4.1 (Tour-factibilidad)**.** *Para cada label ℓ,* $\mathrm{dom}(\ell)$ *puede restringirse a* $\mathrm{dom}(\ell) \cap [0, t]$ *para* $t = \mathrm{mín}\{\mathrm{LDT}(v(\ell), w) \mid w \in V \setminus S(\ell)\}$.

La regla para la dominancia parcial también se obtiene generalizando la dominancia desde el contexto independiente del tiempo. Como se discute en la Sección 4.3, un label independiente del tiempo ℓ es dominado por un label independiente del tiempo m cuando $p(m) + p$ es un tour elemental con una duración no mayor que la duración de $p(\ell) + p$, para cada tour elemental $p(\ell) + p$. En el contexto con dependencia temporal, un label ℓ es dominado por un label m en un tiempo de llegada factible $t \in \mathrm{FAT}(\ell)$ si $t \in \mathrm{FAT}(m)$ y $\ell(t)$ es dominado por $m(t)$. Destacamos que t no necesita pertenecer a $\mathrm{dom}(\ell) \cup \mathrm{dom}(m)$: también puede estar dentro de los intervalos de espera de ℓ o m, que representan intervalos de tiempo donde no ocurre movimiento pero se permite esperar para sincronizar con ventanas de tiempo o mejorar la factibilidad. Como calcular las extensiones factibles de ℓ es costoso, la Regla 4.2 se aplica para descartar labels dominados.

Regla 4.2 (Dominancia parcial)**.** *Un label ℓ es dominado por otro label m en el tiempo $t \in \mathrm{FAT}(\ell)$ si $t \in \mathrm{FAT}(m)$, $v(m) = v(\ell)$, $S(m) = S(\ell)$, y $\Delta_m(t) \leq \Delta_\ell(t)$.*

De ahora en adelante, y para simplificar, cada vez que decimos que ℓ está dominado por m en el tiempo t queremos decir que ℓ y m satisfacen la Regla 4.2. Cada label ℓ mantenido por el algoritmo es tal que ℓ no está dominado en ningún $t \in \mathrm{dom}(\ell)$. Por lo tanto, si un label extendido ℓ está dominado en un tiempo t, entonces t se elimina de $\mathrm{dom}(\ell)$ y ℓ se descarta cuando $\mathrm{dom}(\ell)$ queda vacío. En consecuencia, para $v \in V$ y $S \subseteq V$, si $L = \ell_0, \ldots, \ell_j$ es la secuencia de labels no descartados con $v(\ell_i) = v$ y $S(\ell_i) = S$ tal que $a(\ell_i) \leq a(\ell_{i+1})$ $(0 \leq i < j)$, entonces $b(\ell_i) \leq a(\ell_{i+1})$. De manera coloquial, las duraciones de los labels en L inducen una función parcial sobre $[a(\ell_0), b(\ell_j)]$ (Figura 4.4.3a).

Aunque la regla de bounding de nuestro algoritmo también se basa en las existentes para algoritmos independientes del tiempo, nos resulta más fácil describirla directamente en la terminología dependiente del tiempo. Sea la *duración del tour* $\bar{c}(\ell)$ de un label ℓ la duración mínima de un tour elemental que recorre $p(\ell)$ y visita $v(\ell)$ dentro de su dominio $\mathrm{dom}(\ell)$. Claramente, $\bar{c}(\ell) \geq c^*$ para la duración óptima del tour c^*, y ℓ puede descartarse de forma segura cuando $\bar{c}(\ell) > c^*$. Para calcular la duración del tour de ℓ, observar que un tour $p = (v_0, \ldots, v_{n+1})$ puede visitar v_i $(0 \leq i \leq n+1)$ en el tiempo t si y sólo si t es un tiempo de llegada factible de $p_1 = (v_0, \ldots, v_i)$ y un tiempo de partida factible de $p_2 = (v_i, \ldots, v_{n+1})$; la duración mínima de tal recorrido es $\Gamma_{p_1}(t) + \Delta_{p_2}(t)$. Por lo tanto,

$$\bar{c}(\ell) = \mathrm{mín}\{\Delta_\ell(t) + \Gamma_p(t) \mid t \in \mathrm{dom}(\ell) \cap \mathrm{FDT}(p) \text{ y } p(\ell) + p \text{ es un tour elemental}\}.$$

Al igual que con las reglas anteriores, no sabemos cómo calcular $\bar{c}(\ell)$ de manera eficiente. En su lugar, se utiliza una *cota de finalización* para estimar $\bar{c}(\ell)$ como un proxy. Formalmente, una *cota de finalización cb* es una función que, dado un label ℓ, devuelve una cota inferior de $\bar{c}(\ell)$. Las cotas de finalización se usan junto con una cota superior ub de c^* para descartar labels. Los detalles sobre cómo las calculamos se proporcionan en la Sección 4.6.

Regla 4.3 (Bounding)**.** *Si $cb(\ell) > ub \geq c^*$, entonces $\bar{c}(\ell) > c^*$, por lo tanto ℓ puede ser descartado de forma segura.*

La efectividad de la Regla 4.3 depende fuertemente de la calidad de ub y cb, ya que se descartan más labels cuando ub está cerca de c^* y $cb(\ell)$ está cerca de $\bar{c}(\ell)$. Aun así, la Regla 4.3 puede deshabilitarse tomando $ub = \infty$ y $cb(\ell) = 0$. Sin embargo, nuestro algoritmo requiere un grado mínimo de calidad de cb. Sea $c(\ell)$ la *duración* de ℓ dentro de su dominio, es decir, $c(\ell) = \mathrm{mín}\{\Delta_\ell(t) \mid t \in \mathrm{dom}(\ell)\}$. Decimos que cb es *aceptable* cuando $cb(\ell) \geq c(\ell)$ para cada label ℓ. La razón por la que

(a) Ordenación por buckets *(b)* Fusión

Fig. 4.4.3: (a) Invariante de ordenamiento de un bucket $\mathcal{L}(B) = \ell_1, \ell_2, \ell_3$ a lo largo del Algoritmo 4.4.1. Al incluir las porciones de las piezas de espera representadas en colores más claros, se obtiene una función total sobre $\text{FAT}(\ell_1) \supseteq [a(\ell_1), b(\ell_3)]$. Si los labels rojo y verde se extienden como en (b), entonces su fusión y dominancia en el nuevo bucket, cuya duración previa se muestra en negro, implica tomar el mínimo de dos funciones lineales a trozos, una tarea que requiere tiempo lineal respecto al número de piezas. Aquí, las piezas descartadas están en línea discontinua.

se requiere que cb sea aceptable es porque nuestro algoritmo aplica la Regla 4.4 como condición de terminación. Por simplicidad, a partir de ahora asumimos que toda cota de finalización es aceptable.

Regla 4.4 (Terminación)**.** *Sea ℓ un label que representa un tour elemental y cb una cota de finalización aceptable. Si $cb(\ell) \leq cb(m)$ para todo label m que representa un tour elemental, entonces $p(\ell)$ es óptimo y tiene duración $c^* = cb(\ell)$.*

Nótese que la Regla 4.4 se cumple porque, para cada tour elemental ℓ, tenemos que $cb(\ell) = c(\ell)$.

4.4.3. Implementación

Junto con la instancia de entrada, el algoritmo de labeling (Algoritmo 4.4.1) recibe una cota de finalización cb y una cota superior ub de la duración óptima del tour c^*. A lo largo de su ejecución, el Algoritmo 4.4.1 almacena labels en un diccionario de *buckets* $\mathcal{L}$, donde cada label se complementa con un campo adicional para la cota de finalización descrita anteriormente. Cada label ℓ se inserta en un bucket con índice $(|S(\ell)|, v(\ell), S(\ell))$. Notar que, según la Regla 4.2, ℓ solo puede ser dominado por otro label correspondiente a su mismo bucket. Para cada índice B, $\mathcal{L}(B)$ mantiene una secuencia $\ell_0, \ldots, \ell_j$ de labels, donde $b(\ell_i) \leq a(\ell_{i+1})$ para $0 \leq i < j$. Así, $\Delta(\ell_0) \cup \ldots \cup \Delta(\ell_j)$ define una función parcial sobre el dominio $[a(\ell_0), b(\ell_j)]$, como se muestra en la Figura 4.4.3a. En otras palabras, para cada tiempo t en $[a(\ell_0), b(\ell_j)]$, un bucket mantiene exactamente un label no dominado ℓ_i que tiene a t en su dominio. Cabe destacar que $\mathcal{L}(B)$ contiene todos los labels no dominados con índice de bucket B, por lo que los labels en $\mathcal{L}(B)$ no necesitan representar el mismo camino.

En cada iteración (Pasos 4–15), el algoritmo simultáneamente *procesa* la secuencia L de labels aún no procesados con la mínima cota de finalización b e índice de bucket (k, v, S). Según la Regla 4.4, si $k = |V|$, entonces cada label en L representa un tour con duración c^*. Por lo tanto, el algoritmo se detiene (Paso 8) y ningún label ℓ con $cb(\ell) > c^*$ es procesado. La eficiencia es la razón por la cual todos los labels en L se procesan simultáneamente. De hecho, la *extensión* (Paso 10) es lineal en $|L|$ y en el número de piezas en τ_{vw} (Sección 4.4.1). De manera similar, la dominancia (Paso 13) es lineal en $|M|$ y $|N|$ para la secuencia M obtenida al extender L a través del arco (v, w) y la secuencia N de labels en el bucket de destino. (Figura 4.4.3b), y ocurre un análogo con el bounding (Paso 14). Por lo tanto, buscamos minimizar la cantidad de veces que estas operaciones se ejecutan.

Algoritmo 4.4.1. Algoritmo de labeling Forward para $\mathcal{I} = (D = (V, A), o = 0, d = n + 1, a, b, T, \tau)$.
Entrada: La instancia $\mathcal{I}$, una cota superior ub de c^*, y una cota de finalización cb.
Salida: Si existe, un tour elemental p de duración mínima $c_p \leq ub$.
1: Sea $\ell_0 = (\perp, o, \{o\}, [0, T] \to 0)$ el label inicial, actualizado con $cb(\ell_0)$.
2: Sea q una cola de prioridad de índices no procesados de labels, inicializada con $(cb(\ell_0), 1, o, \{o\})$.
3: Sea $\mathcal{L}$ un mapa de labels procesados inicializado con $(1, o, \{0\}) \to \{\ell_0\}$.

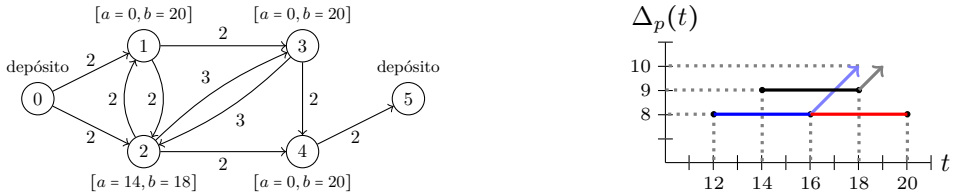
4: **Mientras** $q \neq \emptyset$:
5: Extraer el mínimo (b, k, v, S) en orden lexicográfico de q.
6: Sea L la subsecuencia de labels ℓ en $\mathcal{L}(k, v, S)$ con $cb(\ell) = b$ a procesar.
7: Si $L = \emptyset$, entonces continuar con la siguiente iteración.
8: Si $k = n + 1$, entonces detener y devolver un tour representado por un label en L.
9: **Para cada** $(v, w) \in A$ tal que $w \notin S$:
10: Sea M la secuencia de labels obtenida extendiendo los labels en L.
11: **Para cada** $m \in M$: reducir $\mathrm{dom}(m)$ con la Regla 4.1, y descartar m si $\mathrm{dom}(m) = \emptyset$.
12: Fusionar M en N, donde $N = \mathcal{L}(k + 1, w, S + \{w\})$.
13: **Para cada** $\ell \in N$: aplicar la Regla 4.2 para eliminar todos los tiempos dominados de $\mathrm{dom}(\ell)$.
14: **Para cada** $m \in M \cap N$: actualizar m con $cb(m)$ y descartar m si $cb(m) > ub$.
15: Insertar $(cb(m), k + 1, w, S + \{w\})$ en q para cada $m \in M \cap N$.

Notar que dos labels ℓ y m pueden dominarse mutuamente en diferentes puntos de sus dominios (Figura 4.3.1). Si ℓ se inserta primero en $\mathcal{L}$, entonces ℓ es *corregido* cuando se inserta m (Paso 13), para mantener el invariante del orden de su bucket (Figura 4.4.3b). Como resultado de esta corrección, ℓ se transforma en varios labels, uno por cada segmento no dominado de su dominio (Figura 4.3.1b). Por definición, $cb(\ell') \geq cb(\ell)$ para cada label ℓ' obtenido de esta forma. Sin embargo, dado que recalcular $cb(\ell')$ es un paso costoso, ya que requiere recorrer una estructura de datos como se explica en la Sección 4.6, asignamos $cb(\ell') = cb(\ell)$ asumiendo que estas cotas son cercanas entre sí. Intuitivamente, dado que el dominio de cada ℓ' es más estrecho que el del label original, hay menos posibles extensiones hacia tours factibles. Como resultado, las cotas de finalización permanecen iguales o empeoran. Por lo tanto, las cotas de finalización se calculan únicamente cuando un label es extendido, en el Paso 14.

4.4.4. Relación con el algoritmo independiente del tiempo

Una de las principales diferencias entre el Algoritmo 4.4.1 y el de Tilk et al. [27] es que este último mantiene labels totales y aplica un criterio de dominancia total. Para precisar mejor esta afirmación, suponga que el Algoritmo 4.4.1 se invoca sobre una instancia del MTDP (Figura 4.4.4a), y sea p un camino elemental que comienza en el depósito 0. Como las funciones de tiempo de viaje son constantes, Δ_p tiene un trozo sin espera Δ (Figura 4.4.4b). Por lo tanto, Δ_ℓ es una restricción de Δ, para cualquier label ℓ que representa a p y que es mantenida por el Algoritmo 4.4.1. Por otro lado, tres números son suficientes para codificar cualquier función constante como Δ: los extremos de su dominio y su valor constante. En su algoritmo, Tilk et al. [27] representan p con un label $\hat{\ell} = (\mathrm{prev}, v, S, T_{\mathrm{time}}, T_{\mathrm{dur}}, T_{\mathrm{help}})$ donde: $T_{\mathrm{time}} = \min \mathrm{FAT}(p)$ es el punto inicial del dominio, $T_{\mathrm{dur}} = \Delta(T_{\mathrm{time}})$ es el valor constante de Δ, y $T_{\mathrm{help}} = -\gamma(\max \mathrm{FAT}(p))$ representa el punto final del dominio. Así, $\hat{\ell}$ es un label total en esencia, ya que Δ está codificada en todo su dominio. Además, un label $\hat{\ell}$ se descarta solo si algún label $\hat{m}$ es lexicográficamente menor que $\hat{\ell}$ Tilk et al. [27, Regla 3]. Esto equivale a un criterio de dominancia total en el cual no se consideran los trozos de espera. Como consecuencia de estas decisiones, su algoritmo pierde algunas oportunidades para descartar labels (Figura 4.4.4b).

El Algoritmo 4.4.1 sigue un *bound-ordering*, ya que procesa los labels en el orden dado por cb. En consecuencia, un label ℓ puede ser corregido después de que algunas de sus extensiones hayan sido procesadas. Esas extensiones que surgen de ℓ en los tiempos dominados serán corregidas y posiblemente descartadas. Para evitar procesar tales labels inútiles, una alternativa es aplicar una versión con *bucket-ordering* del Algoritmo 4.4.1 en la que los labels se procesan en el orden dado por sus índices de bucket. En este caso, dado que ℓ solo puede ser corregido por un label en su mismo bucket, ninguna extensión de ℓ se procesa antes de que ℓ sea corregido. Además, todos los labels con el bucket index mínimo se extienden simultáneamente en cada iteración (Pasos 4–15), mejorando la eficiencia general del algoritmo. Para que este algoritmo termine, sin embargo, cada label ℓ con $c^* < cb(\ell) \leq ub$ y $|S(\ell)| < |V|$ debe ser procesado. Por lo tanto, su eficiencia depende de la calidad de ub.



(a) Instancia MTDP *(b)* Duración de los labels

Fig. 4.4.4: (a) una instancia del MTDP. (b) duraciones de las rutas $(0,1,2,3,4)$ (negro), $(0,2,1,3,4)$ (azul) y $(0,1,3,2,4)$ (rojo), incluyendo los tramos de espera. Según la Regla 4.2, el label de la ruta negra se descarta cuando se aplica un dominio parcial. Sin embargo, con dominio total, su label único sobrevive.

Un tercer enfoque, propuesto por Baldacci et al. [9] y aplicado por Tilk et al. [27], es *adivinar* $ub = (1+\varepsilon)cb(\ell_0)$ $(\varepsilon > 0)$ mientras se aplica un ordenamiento por bucket. Si $ub \geq c^*$, entonces el algoritmo encuentra el tour óptimo. De lo contrario, $cb(\ell_0)$ y ub se actualizan a ub y $(1+\varepsilon)ub$, respectivamente, y se ejecuta una nueva iteración. Este enfoque, al igual que el nuestro, es sensible a la calidad de cb. Sin embargo, también depende de $c^* - cb(\ell_0)$ y de la magnitud de ε. Si $(c^* - cb(\ell_0))/\varepsilon$ es grande, entonces se ejecutan muchas iteraciones que calculan repetidamente los mismos labels, mientras que si $(c^* - cb(\ell_0))/\varepsilon$ es pequeño, entonces se examinan muchos labels con $cb > c^*$. Por lo tanto, una buena elección de ε para una instancia puede ser una mala elección para las demás. En este sentido, el Algoritmo 4.4.1 es más robusto porque se adapta a cada instancia, lo cual es importante en nuestro contexto, ya que las cotas calculadas no son tan ajustadas como en problemas independientes del tiempo.

4.5. Técnicas de preprocesamiento

El preprocesamiento está compuesto por tres pasos interrelacionados, que se aplican de forma iterativa hasta que no se produzcan nuevas modificaciones. Para el primero de estos pasos, dirigido a ajustar las ventanas de tiempo, aplicamos las técnicas de Montero et al. [21] y Lera-Romero et al. [17] que extienden las reglas propuestas por Ascheuer et al. [7] para el TSPTW. Debido a limitaciones de espacio, remitimos a los artículos mencionados para una descripción detallada.

El segundo paso implica la inferencia de la relación de precedencia $\prec$ tal que $v \prec w$ cuando v es visitado antes que w en cada tour elemental. Claramente, $0 \prec n+1$ y $0 \prec v \prec n+1$ para $v \in V \setminus \{0, n+1\}$. En la práctica, se infiere un subconjunto de $\prec$ usando la Regla 4.5. Aquí, $\mathrm{EAT}(v,w)$ es el tiempo más temprano en que se puede visitar w cuando se parte de v en el tiempo a_v, mientras que, como en la Sección 4.4.2, $\mathrm{LDT}(v,w)$ es el tiempo más tardío en que se puede partir de v para llegar a w antes de b_w.

Regla 4.5 (Precedencia)**.** *Si* $\mathrm{EAT}(w,v) > b_v$*, entonces* $v \prec w$*. Si* $v \prec z$*,* $\mathrm{EAT}(w,v) > \mathrm{LDT}(v,z)$ *y* $\mathrm{EAT}(v,z) > \mathrm{LDT}(z,w)$*, entonces* $v \prec w \prec z$*.*

El paso final es la eliminación de arcos *infactibles*, es decir, arcos que no pertenecen a ningún tour elemental factible. Claramente, (v,w) es infactible si $w \prec v$ o bien si $a_v + \tau_{vw}(a_v) > b_w$. Nótese que esta última condición no implica que $w \prec v$ porque la desigualdad triangular no necesariamente se cumple.

La Regla 4.5 supone de manera optimista que se puede partir de v en su tiempo de apertura a_v. Sin embargo, durante la enumeración de labels, a veces se llega a v en un tiempo $t > a_v$. Motivados por la formulación ILP de Dash et al. [12] para el TSPTW, donde los intervalos de tiempo pueden preceder a los vértices y viceversa, inferimos precedencias adicionales que son válidas sólo cuando $t > a_v$. Para $v \in V$ y $t \in [0,T]$, sea $\prec_v^t$ la relación tal que $w \prec_v^t z$ cuando w es visitado antes que z en todo tour elemental que visita v en el tiempo t. La Regla 4.6 se aplica para calcular el conjunto de *predecesores* $\pi(v,t) \subseteq \{w \in V \mid w \prec_v^t v\}$ y de *sucesores* $\sigma(v,t) \subseteq \{w \in V \mid v \prec_v^t w\}$.

Regla 4.6 (Precedencia temporal)**.** *Si* $\mathrm{LDT}(v,w) < t$*, entonces* $w \prec_v^t v$ *y* $w \in \pi(v,t)$*, mientras que si* $\mathrm{EAT}(w,v) > t$*, entonces* $v \prec_v^t w$ *y* $w \in \sigma(v,t)$*.*

Claramente, $\pi(v,t) \subseteq \pi(v,t')$ para $t \leq t'$, por lo que existen a lo sumo $n-1$ conjuntos diferentes de predecesores para un $v \in V$ dado. Estos conjuntos pueden precomputarse calculando los puntos de quiebre t que inducen un cambio en $\pi(v,t)$. Luego, $\pi(v,t)$ puede consultarse de manera eficiente para cada $t \in [0,T]$. Los conjuntos $\sigma(v,t)$ se calculan de manera similar. Cabe destacar que la Regla 4.1 codifica implícitamente el uso de $\pi(v,t)$ dentro del Algoritmo 4.4.1. Sin embargo, el propósito del preprocesamiento mejorado es acelerar la ejecución de los algoritmos relajados, como se explica en la siguiente sección.

4.6. Cálculo de cotas de finalización

Recordar que una cota de finalización cb es una función tal que $c(\ell) \leq cb(\ell) \leq \bar{c}(\ell)$ para cada label ℓ, donde $c(\ell)$ y $\bar{c}(\ell)$ son la duración y la duración del tour de ℓ, respectivamente. El algoritmo para calcular $cb(\ell)$ dentro de un algoritmo de labeling que se ejecuta en dirección forward o backward tiene dos pasos. Primero, se ejecuta una adaptación del Algoritmo 4.4.1 para un problema relajado con el fin de obtener un conjunto B de labels no dominados en la dirección opuesta. Luego, se calcula $cb(\ell)$ tomando la duración mínima entre aquellos tours $p(\ell)+p(m)$ para $m \in B$. La corrección de este procedimiento se deriva del hecho de que $\Delta_m(t)$ es una cota inferior de $\Delta_p(t)$ para cada tour $p(\ell)+p$ con un tiempo factible t para visitar $v(\ell)$ (Baldacci et al. [9]).

En este trabajo, se utiliza una relajación *ngP*-tour para calcular B, como se describe en la Sección 4.6.1. Brevemente, el conjunto de tours factibles se amplía permitiendo algunos tours no elementales. Una familia que tiene un vecindario por vértice controla cuáles de los tours no elementales son permitidos por la relajación. Para ajustar las cotas, estos vecindarios se calculan incrementalmente como se discute en la Sección 4.6.2: los vecindarios se amplían en cada paso para definir problemas menos relajados. Para ajustar aún más las cotas, los tours no elementales son penalizados como se explica en la Sección 4.6.3. El vector de penalización óptimo para una relajación dada puede obtenerse resolviendo un LP de gran escala. Este programa difícil se resuelve una vez, antes de construir los vecindarios. Luego, se utiliza el mismo vector de penalización durante todo el proceso, asumiendo que aproxima bien el vector de penalización óptimo de las relajaciones posteriores. Además, dado que el método de penalización es demasiado difícil incluso cuando se considera el escenario menos relajado, se considera una relajación adicional del *ngP*-tour dentro del método de penalización. Esta relajación *ti*-tour, especialmente diseñada para problemas con dependencia temporal, se define en la Sección 4.6.4.

4.6.1. La relajación *ngP*-tour

La relajación *ngP-tour*,[1] propuesta por Baldacci et al. [9] para el TSPTW y adaptada por Tilk et al. [27] para el MTDP, es adecuada para cualquier problema en el que se busca un tour elemental. En resumen, la idea es permitir algunos tours no elementales para reducir el número de buckets almacenados por un algoritmo de labeling. Para mantener las cotas ajustadas, se requiere que los ciclos de estos tours escapen de un conjunto de vecindades predefinidas dentro de las cuales los ciclos de corta duración son comunes. Además, se imponen un gran número de restricciones de precedencia, incluyendo algunas implicadas por aquellos vértices no visitados de tours no elementales, para reducir el número de estados enumerados mientras se mejora la cota inferior.

Recuerde que $v \prec w$ cuando todo tour elemental factible visita v antes que w. Sea $P = (z_1 \prec \ldots \prec z_{|P|})$ una cadena de $\prec$ y suponga que a cada $v \in V$ se le asocia un *vecindario* $N(v) \subseteq V$ que contiene a v. Un ciclo $(v_0, \ldots, v_i)$ es un *ng_N-ciclo* cuando $v_i \in N(v_0) \cap \ldots \cap N(v_i)$. Decimos que un camino p es un *ngP_N-camino* cuando (i) no contiene ng_N-ciclos, (ii) visita cada vértice de P a lo sumo una vez en el mismo orden que P, y (iii) los vértices visitados de P forman un intervalo de $z_1, \ldots, z_{|P|}$. Además, si p es un tour, entonces p es un *ngP_N-tour*. Para maximizar el número de

[1] La relajación es llamada ng$L$-tour por Baldacci et al. [9]; usamos P en su lugar, porque L se usa para conjuntos de labels.

Fig. 4.6.1: Si el *ngP*-tour no elemental $p = (v_0, \ldots, v_{n+1})$ es factible, entonces $v_7 = v_9$ debe ser visitado en un tiempo $t \in [60, 80]$. Entonces, t no respeta π debido a sus predecesores no visitados $p_1, \ldots, p_8$, por lo que p no es *ngP*-factible. De manera similar, el *ngP*-tour no elemental $p = (v_0, \ldots, v_{n+1})$ a la derecha es *ngP*-factible solo si $v_7 = v_9$ puede ser visitado antes del tiempo $t = 60$. Por lo tanto, su *ngP*-duration es una cota superior de su duración.

restricciones impuestas por (ii) y (iii), P se fija como la cadena más larga de la relación derivada de la Regla 4.5. De ahora en adelante, omitimos el subíndice N cuando no haya posibilidad de confusión.

Claramente, todo ngP_M-camino es un ngP_N-camino cuando $N(v) \subseteq M(v)$ para todo $v \in V$, por lo que la duración c de un ngP_N-tour es una cota inferior de la duración óptima del ngP_M-tour. Dado que todo ng_M-tour es elemental cuando $M(v) = V$ para cada $v \in V$, entonces c también es una cota inferior de la duración óptima del tour elemental. En general, vecindarios más grandes producen cotas más ajustadas pero requieren más tiempo computacional. Para mejorar estas cotas, se aplica una noción más fuerte de duración para aprovechar las restricciones de precedencia de los vértices no visitados por un tour no elemental. Sea $p = (v_0, \ldots, v_k)$ un camino con un tiempo de llegada factible t_k, y sea $t_i = \gamma_{(v_i,\ldots,v_k)}(t_k)$, $0 \le i < k$, el tiempo más tardío en el que se puede visitar v_i cuando p visita v_k en el tiempo t_k. Según la Regla 4.6, los conjuntos $\pi(v,t)$ y $\sigma(v,t)$ son conocidos para cada $v \in V$ y $t \in [0,T]$. Recuerde que $w \in \pi(v,t)$ (resp. $w \in \sigma(v,t)$) sólo si w es visitado antes (resp. después) que v en todo tour elemental que visita v en el tiempo t. Decimos que t_k *respeta* π (resp. σ) cuando $|\pi(v_i, t_i)| \le i$ (resp. $|\sigma(v_i, t_i)| \le |V| - i + 1$) para todo $0 \le i \le k$. Un *ngP*-tour p es ng$P$-*factible* cuando contiene un tiempo de llegada factible que respeta π y σ, mientras que su *duración* ng$P$ es la duración mínima en un tiempo de llegada factible que respeta π y σ.

El objetivo en la relajación de *ngP*-tour es encontrar un *ngP*-tour factible de *ngP*con duración mínima. Nótese que si t es un tiempo de llegada factible de un tour elemental, entonces t respeta tanto π como σ. Por lo tanto, la relajación de *ngP*-tour es efectivamente una relajación del TD-TSPTW. Para resolver la relajación de *ngP*-tour, adaptamos el Algoritmo 4.4.1 como se discute en el Apéndice 4.A.3. La Figura 4.6.1 muestra que los tiempos de llegada factibles de un *ngP*-tour general no necesariamente respetan π (o σ), por lo que las duraciones de *ngP*proporcionan cotas más ajustadas que la noción común de duración.

Relación con el problema independiente del tiempo

La relajación *ngP*-tour anterior tiene dos diferencias principales con la relajación *ngP*-tour propuesta por Tilk et al. [27] para el MTDP. En primer lugar, esta última solo verifica que π se respete mediante un tiempo de llegada factible. Según nuestros experimentos, restringir los tiempos factibles a aquellos que también respetan σ ayuda a reducir considerablemente el número de estados enumerados. En segundo lugar, esta última considera las precedencias de manera independiente del tiempo. Es decir, $\pi(v,t)$ se calcula solo para $t = a_v$. En consecuencia, un camino $(v_0, \ldots, v_k)$ se declara factible cuando $|\pi(v_i, a_{v_i})| \le i$ para todo $1 \le i \le k$. Por lo tanto, la ruta más a la derecha en la Figura 4.6.1 se considera factible si cada p_i puede ser alcanzado desde v_7 cuando se parte de v_7 en el tiempo 50. Sin embargo, si v_7 no puede ser alcanzado antes del tiempo 60 cuando se recorre $(v_0, \ldots, v_7)$, entonces la ruta es *ngP*-infactible. Además, si v_7 puede ser alcanzado en el tiempo 50 cuando se recorre $(v_0, \ldots, v_7)$, aún así cada tiempo $t \ge 60$ se declara como *ngP*-infactible, por lo que la *ngP*-duration descarta la llegada a v_7 en el tiempo t como una posibilidad. Por el contrario, la relajación de Tilk et al. [27] utiliza la duración para todo el dominio de tiempos de llegada

factibles como su valor objetivo. En consecuencia, las cotas calculadas por Tilk et al. [27] son más laxas que las proporcionadas por nuestro algoritmo.

Cuando se incorpora al marco general, una diferencia clave es que enumeramos *ngP*-tours, mientras que Tilk et al. [27] enumeran *ngP*2*res*-tours que relajan aún más la factibilidad. Podemos aplicar el primero porque la dominancia parcial, combinada con las nuevas reglas de factibilidad, acelera la ejecución de la relajación de *ngP*-tour, mejorando la relación entre los tiempos de cómputo y la calidad de las cotas inferiores. Los labels en la relajación de *ngP*2*res*-tour se obtienen al descartar T_{time} (Sección 4.4.4), por lo que no codifican el dominio de Δ_ℓ y, en consecuencia, no se puede incorporar la dominancia parcial.

4.6.2. Dynamic Neighbourhood Augmentation

Los vecindarios de los vértices deben definirse antes de que se ejecute la relajación *ngP*-tour. El método de *dynamic neighborhood augementation* (DNA) construye cada vecindario de manera creciente, con el objetivo de "prohibir" *ngP*-tours óptimos que no sean elementales. Si p es un *ngP*-tour óptimo, entonces p es o bien elemental o contiene un ciclo $(v_0, \ldots, v_j)$. En el primer caso, p es una solución óptima para el problema exacto. En el segundo caso, v_0 se inserta en $N(v_i)$, para $0 < i < j$, para "prohibir" $(v_0, \ldots, v_j)$. Este proceso se repite hasta encontrar un tour elemental óptimo o hasta que los vecindarios alcancen un tamaño máximo K. Implementamos el método DNA tal como fue propuesto por Tilk et al. [27], aunque inicializamos $N(v)$ con los cuatro vértices más cercanos a v.

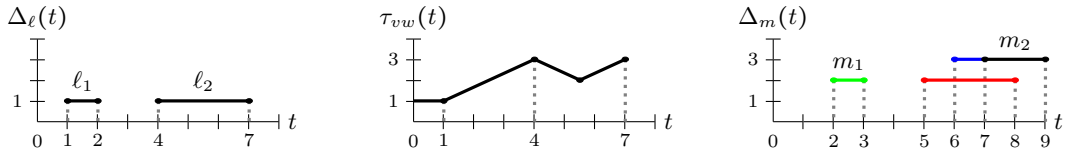
4.6.3. Método de penalización

Las cotas de finalización pueden ajustarse cuando se penalizan los ciclos encontrados en *ngP*-tours. El *método de penalización*, aplicado por Baldacci et al. [9] para el TSPTW y por Tilk et al. [27] para el MTDP, implementa esta idea. Sea Ω el conjunto de tours *ngP*-factibles. Denotemos por $\rho_{vp} \in \mathbb{N}$ el número de veces que el vértice $v \in V$ es visitado por $p \in \Omega$, sea $\nu_v \in \mathbb{R}$ una *penalización* asociada a v, y definamos $\nu_p = \sum_{v \in V}(\rho_{vp} - 1)\nu_v$ como la *penalización* de p. Claramente, $\nu_p = 0$ cuando p es elemental. Por lo tanto, $lb(\nu) = \min\{c_p - \nu_p \mid p \in \Omega\}$ es una cota inferior para el TDTSPTW; aquí c_p es la duración *ngP*de p. Como muestran Baldacci et al. [9], $lb(\nu)$ se maximiza con el vector ν de variables duales correspondientes al problema de programación lineal $\min\left\{\sum_{p\in\Omega} c_p y_p \;\middle|\; \sum_{p\in\Omega} \rho_{vp} y_p = 1 \text{ para } v \in V \text{ y } y_p \geq 0 \text{ para } p \in \Omega\right\}$. En nuestra implementación, resolvemos este problema mediante generación de columnas, adaptando el Algoritmo 4.4.1 (Apéndice 4.A.4). Para una explicación detallada sobre el método de penalización, consulte Baldacci et al. [9].

4.6.4. La relajación *ti*-tour

Cuando el tiempo requerido para resolver el método de penalización hasta la optimalidad supera su límite asignado, ν se obtiene de la última iteración de la generación de columnas. Como discuten Tilk et al. [27], se obtienen mejores penalizaciones cuando se ejecutan más iteraciones de la generación de columnas en un problema más relajado. En las instancias más difíciles, el método de penalización termina antes de alcanzar la optimalidad, incluso cuando se consideran vecindarios pequeños para la relajación *ngP*-tour. En esta sección proponemos una relajación adicional, diseñada específicamente para instancias con dependencia temporal, que permite generar más columnas.

En un contexto con dependencia temporal, mantener varios labels para cada camino factible degrada el rendimiento del algoritmo en cuanto al tiempo de computo. Nuevos labels se crean en dos pasos diferentes: extensión y dominancia. Este último paso tiene un impacto positivo ya que se enumeran menos labels (Sección 4.4.4). Por el contrario, se requiere más tiempo cuando se generan más labels en el primer paso, ya que estos labels deben ser procesados de forma independiente. Sea $\mathcal{I}'$ la instancia MTDP independiente del tiempo obtenida a partir de $\mathcal{I}$ al reemplazar τ_{vw} por $\tau'_{vw} = \min\{\tau_{vw}(t) \mid t \in [a_v, b_v]\}$ para cada arco $(v, w) \in A$. Claramente, $\mathcal{I}'$ es una relajación directa



(a) Duración de los labels. *(b)* Tiempo de viaje τ_{vw}. *(c)* Extensiones vía (v, w).

Fig. 4.6.2: Como $\tau'_{vw} = 1$, los labels verde y rojo en c se obtienen cuando ℓ_1 y ℓ_2 se extienden en la relajación $\mathcal{I}'$. En la *ti*-relaxation, en cambio, $\hat{\tau}^{\ell_1}_{vw} = 1$ y $\hat{\tau}^{\ell_2}_{vw} = 2$, por lo que m_1 y el label azul más el negro se obtienen tras sumar estos tiempos. Sin embargo, como $\tau_{vw}(4) = 3$, la porción azul se descarta de m_2.

de $\mathcal{I}$ en la que se genera un label en cada paso de extensión. Sin embargo, recordamos que se mantienen varios labels por camino cuando se aplica dominancia parcial para resolver $\mathcal{I}'$.

Desafortunadamente, se obtienen cotas pobres al resolver $\mathcal{I}'$ cuando las funciones de tiempo de viaje de $\mathcal{I}$ tienen una alta variabilidad. Además, dado que el tiempo de viaje en la apertura de cada vértice está subestimado, los labels generados al resolver $\mathcal{I}'$ pueden corresponder a *ngP*-tours infactibles. La relajación *ti-tour* que proponemos está diseñada para superar estos problemas (Figura 4.6.2). La relajación *ti*-tour se obtiene ejecutando el Algoritmo 4.4.1, adaptado a la relajación correspondiente de *ngP*-tour, con las siguientes dos modificaciones. Primero, τ_{vw} se reemplaza por la constante (función) $\hat{\tau}^{\ell}_{vw} = \min\{\tau_{vw}(t) \mid t \in \mathrm{dom}(\ell)\}$ cada vez que un label ℓ se extiende mediante un arco (v, w). Así, se obtiene un único label m con una función constante Δ_m. Sólo con esta modificación, el dominio de m sería $[a(\ell) + \hat{\tau}^{\ell}_{vw}, b(\ell) + \hat{\tau}^{\ell}_{vw}]$. Por definición, $\hat{\tau}^{\ell}_{vw} \le \tau_{vw}(a(\ell))$. Por lo tanto, considerando que no se puede alcanzar w antes de $a(\ell) + \tau_{vw}(a(\ell))$ al recorrer $p(\ell)$, la segunda modificación es restringir $\mathrm{dom}(m)$ a $[a(\ell) + \tau_{vw}(a(\ell)), b(\ell) + \hat{\tau}^{\ell}_{vw}]$.

La relajación *ti*-tour tiene al menos tres ventajas interesantes sobre $\mathcal{I}'$ (Figura 4.6.2). Primero, dado que $t' = \arg\min\{\tau_{vw}(t) \mid t \in [a_v, b_v]\}$ no necesariamente pertenece a $\mathrm{dom}(\ell)$, se deduce que $\hat{\tau}^{\ell}_{vw}(t) \ge \tau_{vw}(t') = \tau'_{vw}$ para $t \in \mathrm{dom}(\ell)$ y, en consecuencia, las cotas obtenidas por la relajación *ti*-tour son más ajustadas. Estas cotas se mejoran aún más al recordar que los dominios están restringidos debido la dominancia parcial. Por lo tanto, como segunda ventaja, el uso de $\hat{\tau}^{\ell}_{vw}$ durante el paso de extensión captura parcialmente la variabilidad de los tiempos de viaje: dos labels diferentes ℓ y m extendidos sobre el mismo arco (v, w) considerarán diferentes tiempos de viaje $\hat{\tau}^{\ell}_{vw}$ y $\hat{\tau}^{m}_{vw}$. Finalmente, se sigue por inducción que $a(\ell) \ge \min \mathrm{FAT}(p(\ell))$ para cada label ℓ de la relajación *ti*, es decir, ℓ representa un *ngP*-path factible. En consecuencia, se enumeran menos labels, produciendo cotas más ajustadas, en comparación con $\mathcal{I}'$.

4.7. Algoritmos Desarrollados

Adaptamos el Algoritmo 4.4.1 para desarrollar los métodos LBL, LBL_{ngP} y LBL_{ti}, que reciben una etiqueta $d \in \{fwd, bwd, bi\}$ para indicar si se aplica una búsqueda forward (*fwd*), backward (*bwd*) o bidireccional (*bi*). Los otros parámetros son la instancia $\mathcal{I}$, la cota superior ub, la cota de finalización cb, y los vectores de vecindades N, predecesores π, sucesores σ y penalizaciones ν. Si P es la cadena precomputada de $\prec$ (Sección 4.6.1), entonces:

- $\mathrm{LBL}(\mathcal{I}, ub, cb, \nu)$ resuelve $\mathcal{I}$ con una búsqueda forward que acota con ub y cb,
- $\mathrm{LBL}_{ngP}(\mathcal{I}, d, N, \pi, \sigma, \nu)$ resuelve la relajación ngP_N-tour de $\mathcal{I}$ con una búsqueda en dirección d que no aplica acotamiento y procesa labels con un ordenamiento por bucket, y
- $\mathrm{LBL}_{ti}(\mathcal{I}, N, \pi, \sigma, \nu)$ resuelve la relajación *ti*-tour de la relajación $ng(P, N)$-tour de $\mathcal{I}$ con una búsqueda forward que no aplica acotamiento y procesa labels con un ordenamiento por bucket.

Cada método, a su vez, se adapta para considerar el vector de penalizaciones ν, como en la Sección 4.6.3.

El algoritmo que proponemos para resolver el TDTSPTW, llamado TI, se muestra en el Algoritmo 4.7.1. Su argumento p proporciona una cota superior $ub = c_p$ (Paso 1) y se utiliza para inicializar el método de penalización (Paso 3). Además, el algoritmo se detiene inmediatamente, con salida p, cuando se detecta una cota inferior $lb = ub$. La detención rara vez ocurre antes del Paso 6 porque p se calcula como el primer tour elemental encontrado en una búsqueda en profundidad. Sin embargo, para proporcionar una comparación justa, para el MTDP p es un camino artificial con un costo correspondiente a la cota superior utilizada por Tilk et al. [27] para inicializar su algoritmo.

Algoritmo 4.7.1. TI algorithm for $\mathcal{I} = (D, o, d, a, b, T, \tau)$
Entrada: La instancia $\mathcal{I}$ y un tour elemental p.
Salida: Un tour elemental de duración mínima.
1: Preprocesar $\mathcal{I}$ para obtener π, σ, y P y dejar $ub = c_p$.
2: Para cada $v \in V$, dejar que $N(v)$ contenga a v y los tres vértices más cercanos a v.
3: Sea ν y lb las penalizaciones y cota inferior obtenidas mediante el método de penalización. Cada iteración de pricing se resuelve con $\text{LBL}_{ti}(\mathcal{I}, N, \pi, \sigma, \nu)$, comenzando con $\nu = 0$.
4: Aplicar el método DNA sobre N usando $K = 14$ como tamaño máximo del vecindario. Cada iteración se resuelve con $\text{LBL}_{ngP}(\mathcal{I}, bi, N, \pi, \sigma, \nu)$.
5: Sea cb la cota de finalización obtenida de $\text{LBL}_{ngP}(\mathcal{I}, bwd, N, \pi, \sigma, \nu)$.
6: Devolver el camino dado por $\text{LBL}(\mathcal{I}, ub, cb, \nu)$.

Como los métodos de labeling pueden no terminar en un tiempo razonable, el Algoritmo 4.7.1 impone límites de tiempo a los Pasos 3, 4 y 5+6. La relajación ti-tour no se aplica para el método DNA (Paso 4), ya que se requieren cotas más ajustadas para resolver las instancias más difíciles. Los Pasos 3 y 4 también difieren en el tipo de búsqueda aplicada para el labeling, porque el número de labels Forward enumerados es menor que el número de pares de labels evaluados al hacer la fusión, para los vecindarios pequeños considerados en el Paso 3.

En cada iteración del método DNA, Tilk et al. [27] aplican una búsqueda Forward con una cota de finalización generada a partir de la iteración anterior. La eficiencia de esta variante depende fuertemente de la precisión de ub. Si $ub = \infty$, entonces ningún label es descartado mediante bounding, mientras que si las cotas de finalización están cerca de ub, este enfoque supera en rendimiento a una búsqueda bidireccional. Nosotros evitamos el bounding, incluso para el MTDP, para reducir el impacto de ub. En las instancias más difíciles, donde se ejecuta el Paso 6, el único propósito de ub es descartar labels mediante bounding. Sin embargo, debido al ordenamiento por cotas, ningún label de este tipo es procesado si se considera un ub más laxo. Por lo tanto, ub ayuda a reducir el consumo de memoria pero tiene un impacto menor en la eficiencia temporal. Así, nuestro algoritmo es más robusto ya que depende menos de la precisión de ub.

4.8. Resultados computacionales

Realizamos experimentos computacionales para evaluar el desempeño del Algoritmo 4.7.1 en más de 9000 instancias de referencia, originalmente propuestas para tres problemas diferentes: MTDP, TDTSPTW y TDTSP_m. Incluimos una comparación directa con algoritmos recientes de la literatura para cada caso, con el fin de proporcionar evidencia sustancial sobre la efectividad de nuestro algoritmo. El Algoritmo 4.7.1 está codificado en C++ utilizando CPLEX 12.9 como el solver de LP, y los experimentos se ejecutan en un solo thread en una estación de trabajo con Ubuntu 18.04, un CPU Intel Core i7-8700 a 3.20GHz y 32GB de RAM.

Además del algoritmo TI, los experimentos reportan resultados para los siguientes algoritmos:

- FWD: es simplemente el algoritmo forward sin bounding. FWD proporciona una línea base para evaluar algoritmos de labeling en comparación con los algoritmos BC de última generación.

- NGP: se obtiene reemplazando LBL_{ti} por LBL_{ngP} en el Paso 3 del Algoritmo 4.7.1. NGP evalúa el impacto de la relajación ti-tour dentro del método de penalización.

		FWD		TI						NGP					
n	inst	opt	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
15	400	400	0.03	400	0.49	0.39	0.00	0.44	0.83	400	0.49	0.64	0.00	0.52	1.15
20	400	400	0.45	400	0.84	4.35	0.17	10.00	15.82	400	0.84	7.87	0.17	10.13	19.18
30	400	374	158.74	400	0.83	97.39	0.41	101.57	219.36	400	0.83	188.19	0.40	100.62	309.08
40	400	143	126.55	374	1.35	533.17	0.82	324.21	925.87	350	1.26	743.57	0.86	342.74	1171.92

Tab. 4.8.1: Resultados para el conjunto reducido de instancias TDTSPTW_m.

Los experimentos se organizan de la siguiente manera. Primero, realizamos experimentos preliminares en 1600 instancias de TDTSPTW_m propuestas por Arigliano et al. [5]. Usamos este benchmark reducido para calibrar el algoritmo y establecer una comparación entre FWD, NGP y TI. Posteriormente, comparamos los resultados obtenidos por TI con el mejor enfoque en la literatura para el TDTSPTW_m, el TDTSP_m y el MTDP. Los detalles específicos sobre las instancias y los experimentos se reportan en las secciones correspondientes.

Se establece un límite de tiempo de 3600 segundos para cada instancia con dependencia temporal. Para NGP y TI, el tiempo se distribuye de la siguiente manera: 1200 segundos para el método de penalización (Paso 3), 1200 segundos para el DNA (Paso 2) y 1200 segundos para el labeling exacto (Pasos 5+6), sumando un total de 3600 segundos. Para el MTDP, estos límites se ajustan a 3600 segundos para cada etapa para alinearse con los límites de tiempo usados en los experimentos comparativos. En las tablas, se reportan resultados agregados con las siguientes métricas: inst es el número total de instancias consideradas; opt es el número de instancias resueltas de forma óptima; $\%G_\nu$ es el gap % de la cota inferior obtenida después del método de penalización; T_ν es el tiempo requerido para calcular las penalizaciones; $\%G_N$ es el gap % de la cota inferior obtenida después del DNA; T_N es el tiempo consumido por el DNA; y T_{tot} es el tiempo total de cómputo. El gap % de una cota inferior lb se calcula como $100 \times (z^* - lb)/lb$, donde z^* representa una cota superior para las instancias. Los gaps y tiempos se reportan como el promedio sobre las instancias resueltas de forma óptima por el método correspondiente.

4.8.1. Evaluación de las Relajaciones

Como se reporta en el Apéndice 4.A.5, nuestro método resuelve cada instancia del conjunto de datos utilizado por Vu et al. [28] en menos de dos minutos en promedio. La intuición detrás de este resultado es que todas estas instancias consideran ventanas de tiempo ajustadas, un contexto donde TI y otros enfoques DP son muy efectivos. Por lo tanto, las relajaciones se evalúan en el conjunto de datos TDTSPTW_m de Arigliano et al. [5], que considera ventanas de tiempo más amplias y más períodos de tiempo. Este conjunto de datos tiene 4800 instancias, formadas por 30 instancias por combinación de $n \in \{15, 20, 30, 40\}$, dos escenarios de tráfico, cinco niveles de congestión y cuatro multiplicadores $\beta \in \{0, 0{,}25, 0{,}5, 1\}$ para las ventanas de tiempo, donde valores mayores de β indican ventanas de tiempo más ajustadas.

La Tabla 4.8.1 presenta los resultados para diez instancias seleccionadas aleatoriamente de cada combinación de los parámetros. FWD resuelve todas las instancias con $n \leq 20$ y, en comparación con otros algoritmos en la literatura, una buena proporción con $n \geq 30$. En línea con las variantes independientes del tiempo, tanto NGP como TI obtienen cotas inferiores ajustadas (por debajo del 1 % en promedio) para el algoritmo DP exacto, resolviendo casi todas las instancias en este conjunto reducido. Sin embargo, TI resuelve 24 instancias más que NGP en menos tiempo promedio. Recordar que la diferencia entre estos dos métodos radica en el cálculo de las penalizaciones, donde TI aplica la relajación *ti*-tour. Observar que $\%G_\nu$ es casi el mismo para ambos métodos, incluso cuando las *ti*-tours son una relajación de las *ngP*-tours. Además, la diferencia en los tiempos de ejecución a favor de TI es principalmente consecuencia de la reducción de los tiempos de cómputo en la generación de columnas. En general, este experimento muestra que la relajación *ti*-tour ofrece un buen equilibrio entre los tiempos de cómputo y la calidad de las cotas inferiores. Finalmente, señalamos que el límite de memoria suele alcanzarse en las instancias no resueltas, tanto para TI como para NGP.

		Arigliano et al. [5]		TI					
n	inst	opt	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
15	1200	1185	115.52	1200	0.53	0.38	0.00	0.46	0.85
20	1200	1130	325.17	1200	0.82	4.33	0.17	11.76	18.22
30	1200	744	1169.97	1200	0.87	100.53	0.42	104.38	226.05
40	1200	403	1707.20	1124	1.38	582.80	0.86	361.12	1034.76

Tab. 4.8.2: Resultados para el benchmark de Arigliano et al. [5].

4.8.2. TDTSPTW

La Tabla 4.8.2 compara los resultados de TI con los obtenidos por Arigliano et al. [5] en el conjunto completo de datos para el $TDTSPTW_m$. TI resuelve 4724 de las 4800 instancias, cerrando las instancias con $n \leq 30$ y aumentando el número de instancias resueltas de forma óptima en un 35 % en comparación con el método de Arigliano et al. [5]. A pesar de que se utilizaron diferentes computadoras, los tiempos de ejecución (T_{tot}) se reducen considerablemente con TI, especialmente para $\beta = 1$, donde todas las instancias se resuelven instantáneamente. En general, el comportamiento de TI es consistente con la Tabla 4.8.1: los gaps de optimalidad $\%G_\nu$ y $\%G_N$ son pequeños, con valores ligeramente mayores para $n = 40$, y el método de penalización es el paso más demandante en términos de tiempos de cómputo.

4.8.3. TDTSP

Para evaluar TI en instancias sin ventanas de tiempo, lo comparamos con el algoritmo BB propuesto por Adamo et al. [1], ya que este es el algoritmo más eficiente existente. La Tabla 4.8.3 muestra que nuestro algoritmo resuelve 897 de 1080 instancias, superando las 518 instancias resueltas por Adamo et al. [1]. Los tiempos promedio totales de cómputo son mayores para TI, principalmente porque estos tiempos se calculan sobre las instancias resueltas por el método correspondiente. El tiempo de cómputo aumenta en comparación con las instancias $TDTSPTW_m$ con el mismo valor de n, mientras que los gaps promedio de optimalidad $\%G_\nu$ y $\%G_N$ se mantienen por debajo del 2 %. Este es el comportamiento esperado, ya que el paso de preprocesamiento no tiene efectos debido a la ausencia de ventanas de tiempo. Además, no se infieren precedencias entre vértices, lo que resulta en una regla de factibilidad más débil para la relajación *ngP*-tour. En consecuencia, se enumeran más labels y el algoritmo se vuelve más costoso desde el punto de vista computacional. Asimismo, la proporción relativa del tiempo dedicado por el DNA aumenta, a veces superando el tiempo requerido por la generación de columnas. Las instancias se vuelven más difíciles para $n = 35, 40$, donde en general la generación de columnas y el DNA no convergen dentro del límite de tiempo. En general, el algoritmo TI se posiciona como el mejor enfoque en la literatura hasta ahora para el $TDTSP_m$, siendo capaz de resolver instancias con hasta 40 vértices.

		Adamo et al. [1]		TI					
n	inst	opt	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
15	180	176	240.93	180	1.01	4.47	0.01	32.28	38.11
20	180	141	615.37	178	1.33	36.09	0.42	133.63	220.10
25	180	84	1030.38	164	1.83	220.29	1.02	360.22	778.73
30	180	60	1379.24	151	0.82	802.53	0.41	551.54	1543.86
35	180	35	1426.46	145	0.94	1202.83	0.43	1070.82	2616.96
40	180	22	1278.48	78	1.38	1203.76	0.74	1206.04	2890.87

Tab. 4.8.3: Resultados para el benchmark de Adamo et al. [1].

		Tilk et al. [27]						TI					
dataset	inst	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
Ascheuer et al. [7]	50	50	0.06	51.74	0.03	2.81	54.58	50	0.01	6.49	0.00	1.86	8.40
Gendreau et al. [14]	115	97	0.57	572.87	0.23	25.89	633.07	114	0.19	514.97	0.06	112.59	667.12
Potvin et al. [23]	30	27	2.11	93.09	1.82	11.10	165.38	30	0.60	56.07	0.11	19.01	77.15
Ohlmann et al. [22]	25	1	0.00	729.10	0.00	3.30	732.40	12	0.71	2816.41	0.22	2088.89	5158.68

Tab. 4.8.4: Resultados para el MTDP, benchmark de Tilk et al. [27].

4.8.4. MTDP

Finalmente, evaluamos nuestro enfoque en las instancias MTDP (independientes del tiempo) y comparamos los resultados con los reportados por Tilk et al. [27]. Esta comparación permite medir el impacto de las mejoras incorporadas al algoritmo, a pesar de su adaptación al contexto con dependencia temporal (es decir, dominancia parcial, preprocesamiento, etc.). Las instancias MTDP son instancias bien conocidas de TSPTW, donde se minimiza la duración en lugar de los tiempos de viaje; véase Tilk et al. [27] para una descripción detallada.

Tilk et al. [27] aprovechan el hecho de que las instancias de MTDP utilizan datos enteros, redondeando valores fraccionarios para acelerar la convergencia o, eventualmente, decidir que una solución es óptima. Las soluciones óptimas para el TDTSPTW no necesitan ser enteras, por lo que nuestro algoritmo no aprovecha esta información. De manera similar a Tilk et al. [27], imponemos un límite de tiempo estricto de 10800 segundos para la ejecución total.[2]

El mismo patrón se observa para cada conjunto de datos (Tablas 4.A.2–4.A.5): TI resuelve más instancias, en algunos casos con reducciones significativas —alcanzando el 70 %— en los tiempos de cómputo. En total, se obtuvieron 31 nuevas soluciones óptimas para el conjunto de datos MTDP propuesto por Tilk et al. [27] en menos tiempo de cómputo.

TI reduce consistentemente la $\%G_\nu$. Este efecto puede observarse mejor en el conjunto `Potvin+Bengio`, donde la $\%G_\nu$ siempre está por debajo del 1,5 %. Aunque el nuevo preprocesamiento contribuye en este aspecto, creemos que la diferencia clave es que enumeramos *ngP*-tours en lugar de *ngP*2*res*-tours, como se discute en la Sección 4.6.1. Los tiempos de cómputo a veces se reducen hasta un 30 %, tanto para el método de penalización como para el algoritmo general TI. Estas mejoras se justifican por la discusión anterior y el hecho de que nuestro enfoque enumera menos labels.

4.9. Conclusiones e Investigación Futura

En este capítulo presentamos un algoritmo exacto basado en DP desarrollado principalmente para el TDTSPTW, pero lo suficientemente flexible como para aplicarse a otras variantes. El framework incorpora algunas características de última generación originalmente diseñadas para el TSPTW y el MTDP, que se generalizan y mejoran para manejar funciones de tiempo de viaje y costo más complejas y con dependencia temporal. Desde una perspectiva práctica, los algoritmos eficientes para este tipo de problemas son críticos para obtener mejores planes de distribución en redes congestionadas.

Propusimos distintas contribuciones algorítmicas para explotar la naturaleza con dependencia temporal del problema. Primero, introdujimos nuevas reglas de preprocesamiento que se aplican dentro del algoritmo DP. Segundo, abordamos la dependencia temporal utilizando *labels lineales*, *dominancia parcial* y reglas de factibilidad con dependencia temporal para calcular cotas de finalización ajustadas en un contexto que requiere manejar labels y estructuras de datos más complejas. Tercero, rediseñamos parcialmente el marco independiente del tiempo para hacerlo menos sensible a la disponibilidad de cotas superiores de buena calidad. Finalmente, propusimos la relajación *ti*-tour, que disminuye los tiempos de ejecución mientras alcanza cotas inferiores muy ajustadas para problemas con dependencia temporal.

[2] Según `cpubenchmark.net`, nuestro procesador es aproximadamente un 35 % más rápido que el utilizado por Tilk et al. [27].

Desde un punto de vista práctico, realizamos extensos experimentos computacionales sobre más de 9000 instancias de tres problemas diferentes. En resumen, nuestro algoritmo supera a los enfoques especializados para el TDTSPTW, TDTSP y el MTDP recientemente propuestos en la literatura, proporcionando muchas nuevas soluciones óptimas para instancias previamente no resueltas de los tres problemas. En general, la combinación entre las contribuciones algorítmicas y la sólida evidencia proporcionada por los experimentos sugiere una nueva dirección de investigación hacia algoritmos basados en DP como una de las técnicas de vanguardia para problemas con dependencia temporal.

Creemos que nuestro algoritmo puede motivar varias líneas de investigación. Considerar restricciones operativas adicionales, como recolección y entrega, siempre es interesante. La flexibilidad observada respecto a las ventanas de tiempo hace que el enfoque sea atractivo para problemas más desafiantes, como incluir ventanas de tiempo suaves o variantes consistentes del TDTSP. Finalmente, vale la pena investigar métodos alternativos para calcular cotas de finalización que complementen el enfoque actual, ya que esto representa uno de los ingredientes clave dentro del marco.

Referencias

[1] Tommaso Adamo et al. «An enhanced lower bound for the Time-Dependent Traveling Salesman Problem». En: *Comput. Oper. Res.* 113 (2020), pág. 104795.

[2] David L. Applegate et al. *The Traveling Salesman Problem: A Computational Study.* Princeton, NJ, USA: Princeton University Press, 2007.

[3] Anna Arigliano et al. «A branch-and-bound algorithm for the time-dependent travelling salesman problem». En: *Networks* 72.3 (2018), págs. 382-392.

[4] Anna Arigliano et al. *Time Dependent Traveling Salesman Problem with Time Windows: Properties and an Exact Algorithm.* Inf. téc. 2015.

[5] Anna Arigliano et al. «Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm». En: *Discrete App. Math.* 261 (2018), págs. 28-39.

[6] Norbert Ascheuer et al. «A polyhedral study of the asymmetric traveling salesman problem with time windows». En: *Networks* 36.2 (2000), págs. 69-79.

[7] Norbert Ascheuer et al. «Solving the Asymmetric Travelling Salesman Problem with time windows by branch-and-cut». En: *Math. Program.* 3 (2001), págs. 475-506.

[8] Roberto Baldacci et al. «New route relaxation and pricing strategies for the vehicle routing problem». En: *Oper. Res.* 59.5 (2011), págs. 1269-1283.

[9] Roberto Baldacci et al. «New state-space relaxations for solving the traveling salesman problem with time windows». En: *INFORMS J. Comput.* 24.3 (2012), págs. 356-371.

[10] Nicos Christofides et al. «State-space relaxation procedures for the computation of bounds to routing problems». En: *Networks* 11 (1981), págs. 145-164.

[11] Jean-François Cordeau et al. «Analysis and Branch-and-Cut Algorithm for the Time-Dependent Travelling Salesman Problem». En: *Transport. Sci.* 48.1 (2014), págs. 46-58.

[12] Sanjeeb Dash et al. «A time bucket formulation for the traveling salesman problem with time windows». En: *INFORMS J. Comput.* 24.1 (2012), págs. 132-147.

[13] Yvan Dumas et al. «An Optimal Algorithm for the Traveling Salesman Problem with Time Windows». En: *Oper. Res.* 43.2 (1995), págs. 367-371.

[14] Michel Gendreau et al. «A generalized insertion heuristic for the traveling salesman problem with time windows». En: *Oper. Res.* 46.3 (1998), págs. 330-335.

[15] Michel Gendreau et al. «Time-dependent routing problems: a review». En: *Comput. Oper. Res.* 64 (2015), págs. 189-197.

[16] Soumia Ichoua et al. «Vehicle dispatching with time-dependent travel times». En: *Eur. J. Oper. Res.* 144.2 (2003), págs. 379-396.

[17] Gonzalo Lera-Romero et al. «A branch and cut algorithm for the time-dependent profitable tour problem with resource constraints». En: *European Journal of Operational Research* 289.3 (2021), págs. 879-896. ISSN: 0377-2217.

[18] Gonzalo Lera-Romero et al. «Integer programming formulations for the time-dependent elementary shortest path problem with resource constraints». En: *Electronic Notes in Discrete Mathematics* 69 (2018). Joint EURO/ALIO International Conference 2018 on Applied Combinatorial Optimization (EURO/ALIO 2018), págs. 53-60. ISSN: 1571-0653.

[19] Gonzalo Lera-Romero et al. «Linear edge costs and labeling algorithms: The case of the time-dependent vehicle routing problem with time windows». En: *Networks* 76.1 (2020), págs. 24-53.

[20] Aristide Mingozzi et al. «Dynamic Programming Strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints». En: *Oper. Res.* 45.3 (1997), págs. 365-377.

[21] Agustín Montero et al. «An integer programming approach for the time-dependent traveling salesman problem with time windows». En: *Comput. Oper. Res.* 88 (2017), págs. 280-289.

[22] Jeffrey W Ohlmann et al. «A compressed-annealing heuristic for the traveling salesman problem with time windows». En: *INFORMS J. Comput.* 19.1 (2007), págs. 80-90.

[23] Jean-Yves Potvin et al. «The vehicle routing problem with time windows part II: genetic search». En: *INFORMS J. Comput.* 8.2 (1996), págs. 165-172.

[24] Giovanni Righini et al. «Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints». En: *Discrete Optim.* 3.3 (2006), págs. 255-273.

[25] Martin W. P. Savelsbergh. «The Vehicle Routing Problem with Time Windows: Minimizing Route Duration». En: *ORSA J. Comput.* 4.2 (1992), págs. 146-154.

[26] Peng Sun et al. «The time-dependent capacitated profitable tour problem with time windows and precedence constraints». En: *Eur. J. Oper. Res.* 264.3 (2018), págs. 1058-1073.

[27] Christian Tilk et al. «Dynamic programming for the minimum tour duration problem». En: *Transport. Sci.* 51.2 (2017), págs. 549-565.

[28] Duc Minh Vu et al. «Dynamic Discretization Discovery for Solving the Time-Dependent Traveling Salesman Problem with Time Windows». En: *Transport. Sci.* 54.3 (2020), págs. 703-720.

4.A. Apéndice

En las siguientes secciones describimos las modificaciones requeridas por el Algoritmo 4.4.1 para resolver las diferentes variantes del TDTSPTW discutidas a lo largo del artículo. También se ofrecen algunos otros detalles algorítmicos menores para mayor completitud. Cabe destacar que aquí no se presentan contribuciones científicas, ya que todas las modificaciones requeridas son directas y muchas de ellas ya fueron propuestas en artículos previos. El único propósito, por tanto, es permitir la replicación de los resultados de este capítulo.

4.A.1. El algoritmo de labeling backward

Un *backward label*, que representa un camino $p = (v_0, \dots, v_k)$ que termina en el depósito $v_k = n + 1$, es una tupla $\ell = (\text{next}, v_0, \{v_0, \dots, v_k\}, \Gamma)$, donde next es un puntero a un backward label que representa $(v_1, \dots, v_k)$ y $\Gamma(t)$ es la restricción de un segmento de Γ_p a un intervalo no vacío $\text{dom}(\ell) = [a(\ell), b(\ell)]$. Como en el caso forward, escribimos $v(\ell) = v_0$, $S(\ell) = \{v_0, \dots, v_k\}$, $\Gamma_\ell = \Gamma$, y $\text{FDT}(\ell) = [a(v_0), b(\ell)]$. En lugar de adaptar el Algoritmo 4.4.1, y todas sus variantes, para generar

backward labels, mostramos cómo transformar la instancia para simular el algoritmo backward ejecutando el Algoritmo 4.4.1 tal cual.

Recuerde que una instancia de TDTSPTW es una tupla $\mathcal{I} = (D, o, d, a, b, T, \tau)$, donde o y d representan el depósito inicial y final, respectivamente. La *instancia backward* de $\mathcal{I}$ es la tupla $B(\mathcal{I}) = (D', d, o, a', b', T, \tau')$ tal que D' es el dígrafo obtenido al invertir todos los arcos en D, $a'_v = T - b_v$, $b'_v = T - a_v$, y $\tau'_{wv}(T - t) = t - \max\{t' \in [0, T] \mid t' + \tau_{vw}(t') \leq t\}$ para $t \in [0, T]$. Intuitivamente, los caminos de D son reemplazados por sus reversos, mientras que el horizonte temporal se refleja sobre sí mismo haciendo que el tiempo modificado $T - t \in [0, T]$ represente el tiempo original t. Notar que $B(\mathcal{I})$ es una instancia del TDTSPTW y, además, $B(B(\mathcal{I})) = \mathcal{I}$.

Proposición 4.1. *Si $p = (v_1, \ldots, v_k)$ es un camino de D, entonces $q = (v_k, \ldots, v_1)$ es un camino de D'. Además,* $\mathrm{FDT}(q) = \{T - t \mid t \in \mathrm{FAT}(p)\}$, $\mathrm{FAT}(q) = \{T - t \mid t \in \mathrm{FDT}(q)\}$, *y* $\Delta_q(T - t) = \Delta_p(t)$ *para todo* $t \in \mathrm{FAT}(p)$.

Por la Proposición 4.1, si el Algoritmo 4.4.1 se invoca sobre $B(\mathcal{I})$, entonces cada label Forward ℓ *representa* al label Backward m de $\mathcal{I}$ tal que $\mathrm{next}(m) = \mathrm{prev}(\ell)$, $v(m) = v(\ell)$, $S(m) = S(\ell)$, $\mathrm{dom}(m) = \{T - t \mid t \in \mathrm{dom}(\ell)\}$, y $\Gamma_m(t) = \Delta_\ell(T - t)$ para todo $t \in \mathrm{dom}(m)$. Por lo tanto, la duración obtenida es la misma que cuando el Algoritmo 4.4.1 se ejecuta sobre $\mathcal{I}$. Sin embargo, esta última alternativa es más eficiente cuando se minimiza el makespan. Basta observar que $a(0) = b(0)$, por lo que el dominio de cada label es un punto cuando el Algoritmo 4.4.1 se ejecuta sobre $\mathcal{I}$. En cambio, el label inicial para $B(\mathcal{I})$ tiene un dominio no unitario que se extiende a lo largo del algoritmo.

4.A.2. El algoritmo de labeling bidireccional

Combinamos los algoritmos Forward y Backward para implementar una búsqueda bidireccional (Righini et al. [24]). Por definición, $p(\ell) + p(m)$ es un tour, para un label Forward ℓ y un label Backward m, solo si $|S(\ell)| + |S(m)| = |V| + 1$. En nuestro algoritmo bidireccional, la versión bucket-ordering del Algoritmo 4.4.1 sin bounding se invoca para extender todos los labels Forward y Backward que visitan $\lfloor (|V| + 1)/2 \rfloor$ y $\lceil (|V| + 1)/2) \rceil$ vértices, respectivamente. Así se obtienen dos conjuntos F y B con labels no dominados Forward y Backward, respectivamente. Luego, para cada label $\ell \in F$, se localiza un label $m \in B$ tal que $p(\ell) + p(m)$ es un tour elemental y el costo de fusión $c(\ell, m) = \min\{\Delta_\ell(t) + \Gamma_m(t) \mid a(\ell) \leq t \leq b(m)\}$ es mínimo. Por las Proposiciones 4.2 y 4.3, $p(\ell) + p(m)$ es un tour óptimo para el par (ℓ, m) con costo mínimo $c(\ell, m)$; solo recuerde que $\bar{c}(\ell)$ es la duración mínima entre los caminos que recorren $p(\ell)$ y visitan $v(\ell)$ en un tiempo $t \in \mathrm{dom}(\ell)$.

Proposición 4.2. *Sea ℓ un label Forward, m un label Backward, y $t \in \mathrm{dom}(\ell)$. Entonces, $p(\ell) + p(m)$ es un tour elemental factible en el que $v(\ell)$ puede ser visitado en el tiempo t si y sólo si $v(\ell) = v(m)$, $S(\ell) \cap S(m) = \{v(\ell)\}$, $S(\ell) \cup S(m) = V$, y $a(\ell) \leq t \leq b(m)$. La duración de $p(\ell) + p(m)$ cuando v es visitado en el tiempo t es $\Delta_\ell(t) + \Gamma_m(t)$.*

Proposición 4.3. *Sea ℓ un label forward que visita k vértices. Si B es el conjunto de labels backward obtenidos al ejecutar la versión con bucket-ordering del Algoritmo 4.4.1 hasta que se calculen todos los labels que visitan al menos $|V| - k + 1$ vértices, entonces B contiene un label m tal que la duración de $p(\ell) + p(m)$ es igual a la duración del tour $\bar{c}(\ell)$ de ℓ.*

4.A.3. El algoritmo de labeling para la relajación *ngP*-tour

Recuerde que la relajación *ngP*-tour depende de dos parámetros: P es una cadena más larga de $\prec$ mientras que $N(v)$ es el vecindario de cada vértice $v \in V$. Un camino p es un *ngP*-path cuando no contiene *ng*-cycles, visita cada vértice de P a lo sumo una vez en el mismo orden que P, y los vértices visitados de P forman un intervalo de $z_1, \ldots, z_{|P|}$. Para describir el algoritmo de labeling para la relajación *ngP*-tour necesitamos cambiar la noción de label para codificar *ngP*-paths factibles.

Un *ngP-label* forward de un *ngP*-camino $p = (0 = v_0, \ldots, v_k)$ es una tupla $\ell = (\mathrm{prev}, k, h, v_k, S, \Delta)$, donde prev es un puntero a un *ngP*-label de $(v_0, \ldots, v_{k-1})$, $h = |p \cap P|$ es el número de vértices visitados en P, $S \subseteq V$ es el conjunto de vértices que forman un *ng*-ciclo cuando se insertan después

de v_k, y Δ es la función de duración como se define en la Sección 4.4 para labels. Los *ngP-labels backward* se definen de manera análoga. Para simplicidad, extendemos la notación y terminología de labels a *ngP*-labels, mientras que también definimos $k(\ell) = k$ y $h(\ell) = h$. El *label inicial ngP*es $\ell_0 = (\perp, 1, 1, 0, \{0\}, [0, T] \to 0)$.

Se obtiene un conjunto de *ngP*-labels cuando un *ngP*-label ℓ se extiende a través de un arco $(v(\ell), w)$. Supongamos que $P = (z_1 \prec \ldots \prec z_{|P|})$ y sea $V_h = \{u \in V \mid u \not\succ z_h \text{ y } z_{h+1} \not\prec u\}$ para $0 \le h < |P|$. Claramente, V_h contiene todos los vértices que pueden ser visitados entre z_h y z_{h+1} en cualquier *ngP*-tour factible. Por lo tanto, $p(\ell) + w$ es un *ngP*-path factible solo si $a(\ell) + \tau_{vw}(a(\ell)) \le b_w$ y $w \in V_{h(\ell)} \setminus S(\ell)$. Cuando se cumplen estas condiciones, se crea un label para cada tramo de $\tau_{v(\ell)w}$ que intersecta $\mathrm{dom}(\ell)$, como se explica en la Sección 4.4 para el algoritmo exacto. El cálculo de cada extensión m de ℓ es exactamente el mismo que el descrito en la Sección 4.4 para labels, con la excepción de que $S(m) = (S(\ell) \cup \{w\}) \cap N(w)$, $k(m) = k(\ell) + 1$ y $h(m)$ es igual a $h(\ell)$ o a $h(\ell) + 1$ según si $w \neq z_{h(\ell)+1}$ o no, respectivamente.

Como en el algoritmo exacto, $t \in \mathrm{dom}(m)$ puede ser descartado cuando no existe ningún tour *ngP*-factible $p(m) + p$ que visite w en el tiempo t. Recuerde que $p(m) = (v_0, \ldots, v_k)$ es *ngP*-factible cuando tiene un tiempo de partida factible t_k de $p(m)$ que respeta π y σ, lo que significa que $|\pi(v_i, t_i)| \le i$ y $|\sigma(v_i, t_i)| \le |V| - i + 1$ para todo $0 \le i \le k$, donde $t_i = \gamma_{(v_i,\ldots,v_k)}(t_k)$ es el último tiempo en el que se puede partir de v_i para visitar v_k en el tiempo t_k. Por lo tanto, t puede ser descartado de $\mathrm{dom}(m)$ cuando t no respeta π o σ para el camino $p(m)$. Además, t puede ser descartado cuando z_h no puede ser alcanzado dentro de sus ventanas de tiempo para algún $h > h(\ell)$ o si ya sabemos que ningún tiempo respetará σ cuando se visite $z_{h(\ell)+1}$. Estas restricciones se describen en la siguiente regla; para ver que son correctas, recuerde que $\pi(w, t) \subseteq \pi(w, t')$ y $\sigma(w, t) \supseteq \sigma(w, t')$ para $t \le t'$.

Regla 4.7 (*ngP*-feasibility)**.** *Si m es una extensión de un* ng*P-label ℓ a través de un arco (v, w), entonces* $\mathrm{dom}(m)$ *puede restringirse a $[a, b]$ donde $a = \min\{t \text{ tal que } |\sigma(w, t)| \le |V| - k(\ell) + 1\}$ y $b = \min\{\max\{t \text{ tal que } |\pi(w, t)| \le k(\ell) + 1\}, \min \mathrm{LDT}(w, z_{h(\ell)+1})\}$. Además, si $w \neq z_{h(\ell)+1}$, entonces m puede descartarse si $|\sigma(z_{h(\ell)+1}, b(z_{h(\ell)+1}))| > |V| - k(\ell)$.*

La dominancia también se modifica para manejar la posible repetición de vértices, de la siguiente manera.

Regla 4.8 (*ngP*-dominancia)**.** *Un* ng*P-label m domina a un* ng*P-label $\ell \neq m$ en $t \in \mathrm{dom}(\Delta_\ell)$ si $t \in \mathrm{FAT}(m)$, $k(m) = k(\ell)$, $v(m) = v(\ell)$, $h(m) = h(\ell)$, $S(m) \subseteq S(\ell)$, y $\Delta_m(t) \le \Delta_\ell(t)$.*

El Algoritmo 4.4.1 puede adaptarse para calcular un *ngP*-tour óptimo. Además, se obtiene un conjunto de backward *ngP*(N)-labels no dominados B cuando se ejecuta una versión con ordenamiento por bucket sin bounding en la instancia backward. Supongamos que ℓ es un forward *ngP*(M)-label, para un vector de vecindad M tal que $M(v) \supseteq N(v)$ para $v \in V$. Se dice que $m \in B$ *puede fusionarse* con ℓ cuando $v(\ell) = v(m)$, $S(\ell) \cap S(m) = \{v\}$, $k(\ell) + k(m) = |V| + 1$, y $a(\ell) \le t \le b(m)$. Argumentos similares a los de la Sección 4.A.2 implican que $cb(\ell) = \min\{\Delta_\ell(t) + \Gamma_m(t) \mid t \in \mathrm{dom}(\ell) \text{ y } m \text{ puede fusionarse con } \ell\}$ es una cota de finalización para la relajación del *ngP*(M)-tour. Por lo tanto, un *ngP*-tour óptimo puede calcularse con una búsqueda bidireccional; *cb* es la cota de finalización aplicada por el Algoritmo 4.4.1.

4.A.4. Método de penalización

El objetivo del método de penalización es resolver el problema de programación lineal

$$\min \left\{ \sum_{p \in \Omega} c_p y_p \;\middle|\; \sum_{p \in \Omega} \rho_{vp} y_p = 1 \text{ para } v \in V \text{ y } y_p \ge 0 \text{ para } p \in \Omega \right\},$$

donde Ω es el conjunto de *ngP*-tours factibles *ngP*y ρ_{vp} es el número de veces que $v \in V$ es recorrido por $p \in \Omega$. Resolvemos este problema mediante generación de columnas. El problema de pricing consiste en encontrar un *ngP*-tour $p \in \Omega$ tal que $c_p - \sum_{v \in V} \rho_{vp} \nu_v < 0$, donde ν_v es el valor de la

			Arigliano et al. [5]		TI					
β	n	inst	opt	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
0.00	15	300	287	298.86	300	0.54	0.81	0.00	0.84	1.66
	20	300	248	659.60	300	1.49	9.57	0.41	29.97	45.64
	30	300	155	1631.16	300	1.58	219.72	1.02	217.78	488.56
	40	300	110	2263.33	237	2.27	1115.59	1.62	763.55	2119.87
0.25	15	300	299	143.43	300	0.88	0.54	0.00	0.62	1.17
	20	300	286	382.64	300	1.18	5.55	0.24	13.19	20.92
	30	300	199	1273.53	300	0.97	133.80	0.43	138.34	299.96
	40	300	132	1954.10	287	1.83	924.22	1.07	592.85	1654.10
0.50	15	300	299	25.96	300	0.69	0.17	0.00	0.37	0.55
	20	300	296	289.13	300	0.63	2.19	0.01	3.86	6.32
	30	300	157	1433.36	300	0.94	48.61	0.22	61.39	115.67
	40	300	55	2275.56	300	1.63	418.19	0.90	182.63	619.69
1.00	15	300	300	1.58	300	0.00	0.00	0.00	0.00	0.00
	20	300	300	29.49	300	0.00	0.00	0.00	0.00	0.01
	30	300	233	597.23	300	0.00	0.01	0.00	0.00	0.01
	40	300	106	527.74	300	0.00	0.08	0.00	0.01	0.10

Tab. 4.A.1: Resultados detallados para el benchmark de Arigliano et al. [5].

		Tilk et al. [27]						TI					
type	inst	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
easy	32	32	0.07	3.58	0.04	0.09	3.70	32	0.00	0.94	0.00	1.42	2.37
hard	18	18	0.03	137.37	0.01	7.65	145.03	18	0.02	16.36	0.00	2.64	19.13

Tab. 4.A.2: Resultados detallados para el MTDP, benchmark de Ascheuer et al. [7].

variable dual asociada a la restricción v en el problema maestro restringido. El problema de pricing puede resolverse con algoritmos de labeling similares a los discutidos en la Sección 4.4, adaptados a la relajación en cuestión como se explica en la Sección 4.A.3. En estos algoritmos, se calcula una penalización $\nu_{p(\ell)}$ para cada label ℓ, sumando ν_v por cada vértice visitado v. Luego, se minimiza el costo $\Delta_\ell - \nu_{p(\ell)}$ y, por lo tanto, la Regla 4.8 debe adaptarse para descartar t de $\mathrm{dom}(\ell)$ solo si $\Delta_\ell(t) - \nu_{p(\ell)} \geq \Delta_m(t) - \nu_{p(m)}$.

Para aplicar el método de penalización para el bounding, primero se calcula un conjunto de $ngP(N)$-labels B ejecutando el algoritmo de labeling con un vector de penalizaciones ν, como en el problema de pricing mencionado anteriormente. Luego, se ejecuta un algoritmo de labeling sin penalizaciones en la dirección opuesta. Como observaron Baldacci et al. [9], $cb(\ell) = \min\{\Delta_\ell(t) + \Gamma_m(t) + \nu_{p(m)} \mid t \in \mathrm{dom}(m)$ y m puede ser combinado con $\ell\}$ es una cota de finalización, para cada label ℓ de la segunda ejecución.

4.A.5. Instancias y Resultados Detallados

La Tabla 4.A.1 muestra los resultados de la Tabla 4.8.2 desagregados por el parámetro β, que es un multiplicador para el tamaño de las ventanas de tiempo. La clave de la tabla permanece igual excepto por el parámetro adicional β.

La Tabla 4.8.4 se desagrega en cuatro tablas diferentes, una para cada conjunto de datos de referencia reportado, es decir, la Tabla 4.A.2 para Ascheuer et al. [7], la Tabla 4.A.3 para Gendreau et al. [14], la Tabla 4.A.4 para Potvin et al. [23], y la Tabla 4.A.5 para Ohlmann et al. [22]. Estas tablas mantienen la misma clave, aunque las instancias están agrupadas de diferentes maneras. Las instancias de Ascheuer et al. [7] se separan en fáciles y difíciles como se realizó en Tilk et al. [27]. Las instancias de Gendreau et al. [14] y Ohlmann et al. [22] se agrupan según el número de clientes n. Finalmente, las instancias de Potvin et al. [23] se reportan de forma independiente.

La Tabla 4.A.6 y la Tabla 4.A.7 comparan los algoritmos para el TDTSPTW de Vu et al. [28] (VHBS), de Lera-Romero et al. [18] (LRMB), y nuestro método TI. Estos métodos se comparan

		Tilk et al. [27]						TI					
n	inst	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
20	25	25	1.28	0.57	0.39	0.09	0.68	25	0.23	0.90	0.00	0.08	0.99
40	25	24	0.63	99.24	0.32	16.50	121.50	25	0.23	147.16	0.06	29.00	178.97
60	25	23	0.27	581.29	0.15	16.95	736.87	25	0.16	338.30	0.03	47.82	392.05
80	25	13	0.13	1821.23	0.08	130.25	1951.52	24	0.26	1740.31	0.18	290.59	2165.09
100	15	12	0.01	1343.93	0.01	2.48	1346.41	15	0.01	318.68	0.01	262.53	652.62

Tab. 4.A.3: Resultados detallados para el MTDP, benchmark de Gendreau et al. [14].

		Tilk et al. [27]						TI					
type	inst	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
rc201	4	4	0.06	0.13	0.00	0.03	0.15	4	0.00	0.00	0.00	0.00	0.01
rc202	4	4	0.99	6.58	0.79	0.43	7.10	4	0.24	3.77	0.00	1.98	5.76
rc203	4	4	1.00	92.53	0.79	25.65	119.90	4	0.69	53.49	0.14	42.53	100.92
rc204	3	2	2.01	1001.30	2.01	32.70	1097.50	3	0.52	356.88	0.23	72.00	435.08
rc205	4	4	2.20	5.13	2.19	0.15	5.33	4	0.48	0.24	0.00	0.07	0.32
rc206	4	4	2.79	5.18	1.89	3.40	8.75	4	0.52	5.41	0.00	2.62	8.05
rc207	4	4	5.19	16.20	4.83	18.45	411.88	4	1.10	29.43	0.08	16.91	49.21
rc208	3	1	3.93	7.90	3.32	41.90	57.90	3	1.44	80.71	0.56	32.67	117.43

Tab. 4.A.4: Resultados detallados para el MTDP, benchmark de Potvin et al. [23].

sobre tres conjuntos de datos: *set1*, *w100* y *large*. Los dos primeros conjuntos incluyen instancias con $n = \{15, 20, 30, 40\}$ clientes, y el último con instancias de $n = \{60, 80, 100\}$. Todas las instancias consideran únicamente ventanas de tiempo estrictas. Además, consideramos los dos objetivos (makespan y duración) siempre que la información esté disponible.

Las columnas de estas tablas indican la función objetivo (*obj*), el conjunto de datos considerado (*dataset*), su función objetivo (*obj*) y el número de instancias (*inst*). Además, para todos los métodos, las columnas *opt* y T_{tot} indican el número de instancias resueltas de forma óptima y el tiempo promedio para resolverlas. Finalmente, para el algoritmo TI, se reporta el gap ($\%G_\nu$) y el tiempo de ejecución (T_ν) de la generación de columnas, además del gap ($\%G_N$) y el tiempo de ejecución (T_N) del procedimiento DNA.

		Tilk et al. [27]						TI					
n	inst	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
150	15	1	0.00	729.1	0.00	3.3	732.4	8	0.79	2421.67	0.20	1333.22	4115.00
200	10	0	–	–	–	–	–	4	0.54	3605.90	0.26	3600.23	7246.04

Tab. 4.A.5: Resultados detallados para el MTDP, benchmark de Ohlmann et al. [22].

			LRMB		VHBS		TI					
obj	dataset	inst	opt	T_{tot}	opt	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
makespan	set1	952	950	7.36	952	10.94	952	0.00	0.15	0.00	0.00	0.15
	w100	960	960	21.39	960	1.22	960	0.00	0.10	0.00	0.00	0.10
	large	720	-	-	701	197.42	720	0.00	17.83	0.00	44.07	66.08
duration	set1	952	943	12.87	930	143.45	952	0.00	0.15	0.00	0.00	0.15
	w100	960	957	45.64	955	106.2	960	0.00	0.10	0.00	0.00	0.10

Tab. 4.A.6: Resultados para el TDTSPTW, comparación con Vu et al. [28] (VHBS) y Lera-Romero et al. [17] (LRMB).

		Vu et al. [28]		TI					
n	inst	opt	T_{tot}	opt	$\%G_\nu$	T_ν	$\%G_N$	T_N	T_{tot}
60	240	239	54.26	240	0.00	2.31	0.00	16.83	18.34
80	240	235	239.15	240	0.00	10.45	0.00	25.69	34.88
100	240	227	321.48	240	0.00	40.74	0.00	89.68	145.03

Tab. 4.A.7: Resultados para instancias grandes de Vu et al. [28].

5. ALGORITMO DE BRANCH-CUT-AND-PRICE PARA EL PROBLEMA DE RUTEO DE VEHÍCULOS ELÉCTRICOS CON DEPENDENCIA DEL TIEMPO

5.1. Introducción

Uno de los principales desafíos ambientales en la actualidad radica en el uso de recursos energéticos más limpios y sostenibles para reducir las emisiones y la contaminación a nivel mundial. Proporcionar acceso a nuevas soluciones de transporte sostenibles aparece como uno de los temas urgentes a abordar desde una perspectiva logística. A partir de 2010, se estima que el 20 % de la emisión de gases de efecto invernadero (GHG, por sus siglas en inglés) en esa región provenía de actividades de transporte.

En la última década, muchas empresas han estado cambiando hacia alternativas de transporte más amigables con el medio ambiente. Un área de impacto significativo es la logística de última milla, donde los vehículos tradicionales con motor de combustión interna (ICEVs, por sus siglas en inglés) están siendo reemplazados por vehículos eléctricos (EVs, por sus siglas en inglés) y bicicletas de carga, especialmente en áreas altamente congestionadas. Por ejemplo, UPS estableció como objetivo que el 25 % de sus vehículos funcionen con combustible alternativo para 2020, incluyendo 10,000 EVs UPS [43]. Desde una perspectiva técnica, los EVs ofrecen varios beneficios en comparación con los ICEVs, como menor contaminación acústica, reducción de emisiones de CO_2 y una alta eficiencia en la conversión de energía. Usualmente se combinan con otros métodos como el sistema de frenado regenerativo, que produce energía eléctrica a partir del movimiento. Como contraparte, la capacidad limitada de la batería influye en las operaciones de los EVs al reducir el llamado *rango de conducción* (driving range). Por lo tanto, los planes de ruteo para EVs deben incorporar estas nuevas restricciones operativas explícitamente, evitando que los vehículos se queden sin batería durante la distribución mediante paradas intermedias en estaciones de recarga. Los avances recientes han incrementado el rango de conducción en los últimos años, y las implementaciones actuales sugieren que está alcanzando niveles razonables para la distribución dentro de las ciudades. Sin embargo, la capacidad de la batería disminuye con el tiempo y sus reemplazos continuos representan inversiones significativas. Otros aspectos como el uso de sistemas de aire acondicionado o calefacción también reducen el rango de conducción, que puede ser hasta un 30 % de la capacidad original según Restrepo et al. [31].

El VRP con Vehículos Eléctricos (EVRP, por sus siglas en inglés) introducido por Schneider et al. [38] extiende el VRP incorporando explícitamente la batería del vehículo. Se modela como un recurso adicional donde la energía se consume cuando el vehículo se desplaza a lo largo de la red y puede recargarse en ruta. La mayor parte de la literatura considera un modelo simplificado donde la descarga de la batería es proporcional a la distancia recorrida, asumiendo que la tasa de consumo permanece constante durante el horizonte de planificación. Una observación similar se aplica a los tiempos de recarga en las estaciones de carga. En la práctica, según Goeke et al. [15] y como se observa en Schneider et al. [38], la carga del vehículo, la velocidad de viaje y la pendiente del terreno están entre las variables más importantes que afectan el consumo de energía.

Estos parámetros se vuelven relevantes en la práctica al implementar los planes de ruteo. Por ejemplo, dentro de la logística de última milla en grandes ciudades, las variaciones del tráfico impactan directamente en las velocidades de viaje en diferentes momentos del día, lo que a su vez afecta la descarga de la batería. En este contexto, los paquetes a distribuir son usualmente bastante pequeños y la carga no es tan importante como las decisiones de tiempo. Si la autonomía representa una limitación, ya sea por la tecnología disponible o la degradación de la batería, usar la distancia como un proxy para el consumo de energía puede ser engañoso. Otra área relevante se refiere a la logística de media distancia, donde la transición a una flota electrificada incluye algunos desafíos adicionales en comparación con las operaciones de última milla, como se discute

en Schiffer et al. [35]. Una ruta estándar de media distancia naturalmente excede la autonomía de los EVs actuales, haciendo que las operaciones sean más complejas ya que las recargas en ruta no pueden evitarse. Los efectos de congestión también juegan un papel importante en este caso, ya que, dependiendo de la ubicación geográfica de los depósitos, centros de distribución y tiendas, los tiempos y velocidades de viaje pueden verse fuertemente afectados durante las horas pico, especialmente si se consideran grandes áreas urbanas. Reconocemos que la carga total del camión, y cómo esto impacta las operaciones, también sería un factor importante a evaluar, no incluido como parte de nuestra investigación. En general, estos dos escenarios indican que los niveles de batería en ruta podrían ser estimados erróneamente bajo el modelo clásico de descarga lineal, conduciendo a planes tácticos que podrían no ser aplicables en la práctica debido a la descarga de la batería.

Aunque la congestión es un fenómeno dinámico, donde los eventos inesperados pueden ser difíciles de predecir en la planificación con un día de anticipación, diseñar planes de distribución más realistas sigue siendo un área de estudio relevante. Se espera que los planes de ruteo para entregas del día siguiente que consideren los patrones promedio de congestión se traduzcan en operaciones mejoradas, posiblemente con menos interrupciones debido a variaciones de tráfico predecibles. Nuestro artículo contribuye en esta dirección. Estudiamos el EVRPTW con dependencia temporal (TDEVRPTW, por sus siglas en inglés), que incorpora los efectos de la congestión en el EVRPTW originalmente propuesto en Schneider et al. [38], no sólo en términos de la sincronización sino también en la carga de la batería. Además, proporcionamos un marco general que integra de forma natural otros aspectos operativos relevantes que afectan la aplicabilidad de los planes de ruteo, tales como tiempos variables de carga y tiempos de espera en las estaciones de recarga. Primero ofrecemos una discusión sobre la literatura relacionada con el EVRPTW y el TDVRP para enmarcar nuestra investigación, y luego describimos nuestras principales contribuciones.

5.1.1. Revisión de la literatura

Como se describe en la siguiente sección, nuestro artículo contribuye introduciendo el TDEVRPTW tanto en términos del modelado como en el desarrollo de un algoritmo de BCP de última generación como metodología de resolución. En cuanto a algoritmos exactos, los BCP son los algoritmos más efectivos, especialmente para VRPs con múltiples vehículos. Remitimos al lector a Toth et al. [42] para una introducción detallada a los algoritmos BCP aplicados a VRPs, y exploramos además Irnich et al. [20] y Irnich [19] respecto a algoritmos avanzados de labeling para problemas de camino mínimo con restricciones de recursos, incluyendo el uso de Resource Extension Functions (REFs). Más recientemente, Costa et al. [3] ofrecen una visión completa de los diferentes componentes que usualmente se incorporan en un BCP.

Los aspectos ambientales dentro de los VRP han sido abordados desde diferentes perspectivas. Una línea de investigación relacionada con *green logistics* considera, entre otros, los efectos del transporte sobre el medio ambiente mediante la minimización del consumo de combustible y las emisiones de GHG bajo diferentes condiciones de congestión (véase, por ejemplo, Heni [16] y Toth et al. [42, Capítulo 15]). Según nuestro conocimiento, el primer enfoque que incluye vehículos con combustible alternativo y un rango de conducción limitado es el Green VRP (GVRP) propuesto por Erdoğan et al. [10]. Aunque no se enfocan en el uso de EVs, proporcionan un marco inicial para este tipo de problemas. Davis et al. [5] estudian el impacto de reemplazar ICEVs por EVs para distribución y Felipe et al. [11] extienden el GVRP introduciendo la política de recarga parcial, permitiendo que los vehículos determinen la carga de batería en cada estación de recarga, así como múltiples tecnologías de recarga asignando una velocidad de recarga diferente a cada estación.

Recientemente, una serie de artículos considera explícitamente problemas de distribución utilizando una flota de vehículos eléctricos, donde la energía de la batería del vehículo se consume a medida que el vehículo opera, y puede ser recargada para eventualmente extender la duración de la ruta. El EVRP con ventanas de tiempo (EVRPTW, por sus siglas en inglés) fue propuesto en Schneider et al. [38], donde el GVRP se extiende con restricciones limitadas de batería y otras restricciones clásicas del VRP (por ejemplo, ventanas de tiempo y capacidades de vehículos), y se aborda mediante metaheurísticas. Se asume que la descarga de la batería es lineal con respecto a

la distancia recorrida, una simplificación de las condiciones reales. Desaulniers et al. [7] desarrollan un algoritmo BCP para una generalización del EVRPTW donde se permiten políticas de recarga completas y parciales, así como la posibilidad de limitar el número de recargas en ruta. De manera similar, Roberti et al. [32] abordan la versión de un sólo vehículo del problema. Las recargas parciales también se consideran en Keskin et al. [23], mientras que Schiffer et al. [36] incorpora además decisiones de ubicación de las estaciones. Cabe destacar que estos problemas pueden formularse como un caso especial del VRP con paradas intermedias, una familia más general de VRPs recientemente revisada en Schiffer et al. [37].

Otra área de investigación interesante y complementaria se centra en enriquecer el EVRPTW y otras variantes incorporando características para reducir la brecha con las operaciones del mundo real. Demir et al. [6] presentan una revisión de diferentes modelos de consumo de energía. Modelos de consumo de energía más complejos y realistas para baterías eléctricas, basados en pruebas realizadas con datos reales, son propuestos por Goeke et al. [15] y Wu et al. [44]. Cabe destacar que Goeke et al. [15] extienden el EVRPTW para considerar una flota mixta compuesta tanto por EVs como por ICEVs. Su modelo incorpora la dependencia del consumo de energía respecto a la velocidad de viaje, la masa del vehículo y el peso de la carga a transportar, entre otros. Resultados adicionales interesantes son reportados por Fetene et al. [12], donde se mide el efecto de diferentes factores en la tasa de consumo de energía y la autonomía mediante un análisis detallado de un conjunto de datos a gran escala de patrones de conducción recopilados de más de 700 EVs durante un período de dos años. Los EVs corresponden a vehículos privados en lugar de vehículos pesados, y se consideran varios factores, como la velocidad de viaje, el comportamiento del conductor, las condiciones climáticas, y el tipo de carretera entre otros, al analizar su uso. En cuanto a la velocidad de viaje, sus resultados sugieren que el consumo de energía es no lineal y que la velocidad de conducción óptima tiene un punto óptimo (*sweet spot*) entre 45 y 56 km/h, inferior a los 65 km/h reportados para los ICEVs. Al comparar las características de cada modelo, se observan diferencias en el consumo de energía, particularmente a velocidades de viaje más bajas. Esto sugiere que se requiere investigación adicional en esta área para afinar los modelos de consumo de energía.

Recientemente, y de manera simultánea con nuestro trabajo, algunos enfoques investigan algoritmos para funciones de descarga de batería más complejas. Zang et al. [45] consideran un modelo no lineal de depreciación de la batería para tener en cuenta el impacto de la *profundidad de descarga* y desarrollan una heurística basada en generación de columnas con un algoritmo de labeling adaptado. Rastani et al. [30] abordan una cuestión de investigación similar a la nuestra pero centrada en el peso. Ajustan el modelo de consumo de Goeke et al. [15] para considerar la carga, proponen dos formulaciones ILP compactas y proponen una heurística Adaptive Large Neighborhood Search (ALNS) para investigar el impacto de la carga en el consumo de energía.

La interacción entre los vehículos y la infraestructura de recarga introduce nuevas e interesantes restricciones operativas, con un impacto directo en la calidad y la factibilidad de las soluciones. Estas restricciones están motivadas por contextos prácticos y, en general, han sido abordadas de forma independiente. Sassi et al. [33] consideran costos de recarga dependientes del tiempo, múltiples tecnologías de recarga y una flota mixta de vehículos, modelando las estrategias dinámicas de precios en redes de smart grid. Bessi et al. [1] desarrollan un algoritmo BP considerando múltiples tecnologías de recarga y proponen un algoritmo de labeling bidireccional, asumiendo aún que el tiempo de recarga es proporcional a la cantidad de energía recargada. Montoya et al. [27] se enfocan en el proceso de recarga, observando que la cantidad de energía cargada depende no sólo del tiempo empleado sino también del nivel preliminar de energía de la batería. Las funciones de recarga que modelan el proceso son no lineales, y muestran que funciones lineales a trozos continuas con hasta 3 segmentos constituyen buenas aproximaciones. Este problema también es estudiado por Froger et al. [13], introduciendo nuevos algoritmos que mejoran la calidad de las soluciones. Otro aspecto interesante se relaciona con el acceso a la infraestructura de recarga. La capacidad limitada de una estación de recarga (en términos de cargadores) puede ser superada por la alta demanda de vehículos durante las horas pico. Esto es considerado en Keskin et al. [22], donde los vehículos no pueden asumir que tienen acceso inmediato a un cargador al llegar a una estación, y por lo tanto incurren en tiempos de espera que afectan el resto de la ruta. Sus experimentos

sugieren que los tiempos de espera pueden resultar en hasta un 26 % de diferencia en términos de costos. De manera similar a los casos anteriores, los tiempos de espera pueden ser modelados mediante funciones lineales a trozos continuas que cumplen la condición First-In-First-Out (FIFO). En general, remitimos al lector a Pelletier et al. [28] para una revisión completa de la literatura que considera aspectos de modelado y algorítmicos.

Los efectos de la congestión en las decisiones de ruteo han recibido una atención creciente durante los últimos años, ya que representan un aspecto crítico respecto a las entregas de última milla en grandes ciudades (véase por ejemplo Savelsbergh et al. [34]). Tal como se describe en el Capítulo 1 que Time-Dependent VRPs es el nombre asignado a una amplia familia de problemas de optimización interesantes que incorporan explícitamente las condiciones del tráfico, asumiendo que el tiempo de viaje entre dos clientes es variable durante el horizonte de planificación. Respecto al TDVRP, Dabia et al. [4] desarrollan un algoritmo BP para el TDVRP con ventanas de tiempo donde la función objetivo minimiza la duración total de las rutas. El problema de pricing se aborda utilizando algoritmos de labeling adaptados que incorporan la dependencia temporal. Como resultado, algunos de los recursos se modelan mediante una función que depende del tiempo. El algoritmo de labeling aplica un *criterio de dominancia total*, lo que significa que un label puede dominar a otro solo si esta función está dominada en todo su dominio. El TDVRPTW con Recolecciones y Entregas (TDVRPTWPD, por sus siglas en inglés) es estudiado en detalle por Sun et al. [40] usando ideas similares, mientras que una variante del TDVRPTW con flexibilidad de ruta es explorada por Huang et al. [17]. Otro problema con una estructura similar es estudiado en Tagmouti et al. [41], donde un problema de ruteo por arcos con costos de servicio lineales a trozos dependientes del tiempo se transforma en un VRP, considerando también una función de servicio dependiente del tiempo. Respecto al enfoque de solución, el algoritmo de labeling desarrollado para el problema de pricing se asemeja al propuesto en Dabia et al. [4], que considera un criterio de dominancia total. Recientemente, Lera-Romero et al. [25] abordan el TDVRPTW con un algoritmo BCP que incorpora dominancia parcial dentro del algoritmo de labeling al resolver el problema de pricing, permitiendo que un label sea dominado solo en una porción de su dominio. Esta mejora demuestra ser muy efectiva, superando enfoques previos en los experimentos computacionales.

La literatura que conecta el EVRPTW y el TDVRP es algo escasa, y destacamos algunas investigaciones recientes en esta dirección. Fukasawa et al. [14] estudia un problema de optimización complejo donde, además de las decisiones de ruteo, la velocidad de los vehículos en cada instante también es una variable de decisión con el fin de minimizar los costos asociados al consumo de combustible. Pelletier et al. [29] consideran un problema de optimización bajo la incertidumbre de algunas variables externas como la fricción de la carretera, la velocidad del vehículo y el clima, estimando la desviación de la tasa de descarga en escenarios típicos. Shao et al. [39] mejoran el EVRPTW clásico añadiendo tiempos de viaje dependientes del tiempo para obtener estimaciones más precisas respecto a posibles violaciones de las ventanas de tiempo, aunque el modelo de descarga de la batería permanece simplificado y no se ve afectado por las velocidades variables de viaje. Recientemente, y simultáneamente con nuestro trabajo, Lu et al. [26] dan los primeros pasos hacia un modelo que considera el impacto de las velocidades dependientes del tiempo en el consumo de energía. Sin embargo, asumen que la congestión puede ser capturada con una función escalonada que tiene solo tres tramos con diferentes velocidades de viaje. Más allá de las limitaciones respecto a la aplicabilidad práctica, el modelo propuesto no escala para contextos más generales. Hasta donde sabemos, ningún artículo en la literatura ha propuesto un modelo general que integre los efectos de la congestión en el modelo de descarga de batería para vehículos eléctricos.

5.1.2. Nuestras Contribuciones

Nos basamos en la investigación de Goeke et al. [15] y Ichoua et al. [18] para estudiar el TDEVRPTW, una generalización del EVRPTW que captura el efecto de la congestión tanto en los tiempos de viaje como en el consumo de batería. En línea con las observaciones planteadas por Desaulniers et al. [7], considerar explícitamente los perfiles de velocidad como entrada para el modelo de descarga de batería introduce una complejidad adicional respecto a las REFs en comparación con el EVRPTW. Aunque la velocidad no es en sí misma una variable de decisión, las

rutas deben encontrar implícitamente un equilibrio entre viajar por un arco en un período menos congestionado para reducir el tiempo de viaje, lo cual probablemente sea beneficioso respecto a las ventanas de tiempo, y el eventual aumento en el consumo de energía, que posiblemente afecte negativamente la autonomía de conducción.

Sin embargo, nuestra concepción de la dependencia temporal va más allá de los tiempos de viaje clásicos con dependencia temporal, ya que proponemos un marco unificador que integra de manera natural otras restricciones operativas como los tiempos variables de carga (Montoya et al. [27]) y los tiempos de espera (Keskin et al. [22]), ambos aproximados mediante funciones lineales a trozos continuas en sus enfoques. Hasta donde sabemos, nuestro artículo es el primero en considerar todos estos aspectos de manera integrada desde la perspectiva del ruteo. A través de nuestro entorno experimental, proporcionamos valiosos insights de negocios sobre el impacto de la congestión en el rango de conducción. Para algunos de los escenarios considerados, hasta un 40 % de las rutas obtenidas en un entorno sin dependencia temporal se vuelven infactibles debido al impacto de la congestión en la batería.

En términos de metodología, diseñamos un algoritmo BCP que incluye componentes de última generación. Desarrollamos un algoritmo de labeling capaz de manejar funciones lineales a trozos dentro de cada label para codificar el consumo de energía dependiente de la velocidad como un recurso. Nuestro algoritmo de labeling aplica reglas de dominancia parcial para acelerar su ejecución. El BCP se mejora con nuevas reglas de preprocesamiento específicas para el TDEVRPTW que pueden ser aprovechadas además por otras variantes de VRP que consideran un rango de conducción limitado. En cuanto al branching, proponemos un esquema alternativo para el ruteo con EVs que simplifica el propuesto en Desaulniers et al. [7] y, además, reduce el número de nodos enumerados en un BP. Experimentos computacionales extensos demuestran que nuestro enfoque es robusto para las instancias con dependencia temporal, que nuestro framework más general es eficiente en benchmarks de EVRPTW sin dependencia temporal, y que es capaz de resolver instancias con dependencia temporal de tamaño similar a las del EVRPTW con un esfuerzo computacional comparable.

El resto del capítulo está organizado de la siguiente manera. La Sección 5.2 define el TDEVRPTW en su forma general e introduce la notación utilizada en el capítulo. Luego, la Sección 5.3 presenta un marco general que codifica dentro del TDEVRPTW restricciones operativas adicionales relacionadas con el tiempo. El algoritmo BCP se presenta en la Sección 5.4. Finalmente, los experimentos numéricos se presentan en la Sección 5.6 con conclusiones y una discusión sobre posibles direcciones futuras en la Sección 5.7.

5.2. Problema de ruteo de vehículos eléctricos con tiempos de viaje variables

En esta sección definimos el TDEVRPTW en su forma general y ofrecemos un breve análisis de sus características novedosas. Revisamos las variantes de la literatura que se incorporan posteriormente en los algoritmos descritos en la Sección 5.4.2. Finalmente, presentamos el nuevo modelo de consumo de batería con dependencia temporal, que integra las velocidades variables en las funciones de descarga de batería.

5.2.1. Definición del problema

El TDEVRPTW se define sobre un grafo dirigido $D = (V, A)$ donde cada vértice $i \in V$ representa una ubicación que puede ser el depósito, un cliente o una estación de recarga. Así, $V = \{o, d\} \cup V_c \cup V_s$, donde o y d son dos vértices que representan el depósito, V_c es el conjunto de clientes, y V_s el conjunto de estaciones de recarga. Cada arco $(i, j) \in A$ representa un camino entre las ubicaciones i y j, donde c_{ij} denota su costo de viaje.

Los problemas clásicos de ruteo incorporan restricciones operativas de la vida real, como la capacidad del vehículo, ventanas de tiempo y tiempos de servicio. Suponemos que está disponible una flota ilimitada de EVs homogéneos, donde cada vehículo tiene una capacidad Q. Cada cliente

$i \in V$ tiene una demanda positiva q_i, una ventana de tiempo $[a_i, b_i]$ en la que debe comenzar el servicio, y un tiempo de servicio s_i para atender al cliente.

Los efectos de congestión se capturan siguiendo el modelo propuesto por Ichoua et al. [18]. Recordar de la Sección 1.1 que las operaciones tienen lugar dentro de un horizonte de planificación $[0, T]$. Cada arco $(i, j) \in A$ tiene una distancia asociada d_{ij} y una *función de velocidad* $v_{ij}(t)$ que indica la velocidad de cualquier vehículo que recorra el arco (i, j) en el tiempo t. En este modelo, $v_{ij}(t)$ es una función escalonada sobre una partición del horizonte de planificación dada como entrada. Luego, se deriva una *función de tiempo de viaje* $\tau_{ij}(t)$ para cada arco $(i, j) \in A$, que indica el tiempo de viaje del trayecto de i a j si se parte en el tiempo $t \in [0, T]$. La función $\tau_{ij}(t) : [0, T] \to \mathbb{R}$ es continua, lineal a trozos, y satisface la propiedad FIFO, lo que significa que demorar la partida no puede conducir a un tiempo de llegada más temprano.

Cada EV está equipado con una batería con una capacidad limitada B (expresada en kWh). Para cada arco $(i, j) \in A$ consideramos una *función de consumo de energía* $\beta_{ij}(t)$ que indica la energía consumida al viajar desde la ubicación i hasta j si se parte de i en el tiempo t. Como se describe en la siguiente sección, el consumo depende de la velocidad de viaje y, por lo tanto, se convierte en una función con dependencia temporal sobre el tiempo de partida desde la ubicación de origen. Aunque la batería se descarga mientras el vehículo está en movimiento, puede recargarse en cualquier estación para continuar la ruta. Para un EV con un nivel inicial de batería de $w_0 > 0$ kWh y una estación de recarga $j \in V_s$, la carga obtenida después de t unidades de tiempo está dada por una función $g_j(w_0, t)$. Siguiendo la notación introducida por Desaulniers et al. [7], el EVRPTW clásico se recupera al establecer la función de descarga de batería $\beta_{ij}(t) = b_{ij}$ y la función de recarga $g_j(0, h_{ij}) = b_{ij}$ cuando se asumen funciones de tiempo de viaje constantes, siendo b_{ij} el consumo de energía para el arco $(i, j) \in A$ y $h_{ij} = \alpha b_{ij}$ el tiempo requerido para recargar b_{ij} unidades de energía. La Sección 5.2.2 proporciona un análisis más profundo de estas funciones.

La mayor parte de la literatura relacionada con los EVs permite recargas de una de dos políticas: completa o parcial. Mientras que la política de *recarga completa* obliga a todos los vehículos a cargar las baterías hasta su capacidad total en cada estación, el escenario de *recarga parcial* ofrece una opción más detallada. La recarga parcial amplía el espacio de soluciones al indicar no sólo la ruta sino también la cantidad de batería recargada en cada estación visitada dentro de la ruta. Desaulniers et al. [7] también limita el número de recargas en ruta a una sola. Esta política se denomina *política de recarga única*, mientras que la anterior se conoce como *política de recarga múltiple*.

Una ruta $r = (0, v_1, \cdots, v_k, d)$ es una secuencia de ubicaciones que comienza en el depósito inicial o y termina en el depósito final d. De manera similar al EVRPTW, una ruta r es *factible* si y sólo si: (i) no hay clientes repetidos (aunque una estación de recarga puede ser visitada más de una vez); (ii) no se excede la capacidad del vehículo; (iii) los clientes son visitados dentro de su ventana de tiempo $[a_i, b_i]$; y (iv) la energía requerida $\beta_{ij}(t)$ para cada arco (i, j) en una ruta nunca debe exceder el nivel de batería del EV al salir de la ubicación i hacia j. Las dos últimas condiciones dependen del tiempo de la ruta, ya que tanto el tiempo de viaje como el consumo de batería dependen de la velocidad que, a su vez, depende del tiempo de partida desde cada uno de los vértices recorridos. Entonces, como se discute en las siguientes secciones, los tiempos de partida desde el depósito y desde las ubicaciones intermedias dentro de una ruta se convierten en una variable de decisión. El costo de una ruta r está dado por $c_r = \sum_{(i,j)\in r} c_{ij}$. El TDEVRPTW consiste en encontrar un conjunto de rutas factibles que visiten a cada cliente exactamente una vez al mínimo costo total.

Además, señalamos que, aunque la mayoría de los modelos con dependencia temporal tienen una función objetivo que minimiza el makespan o la duración de las rutas, consideramos una función objetivo más clásica e independiente del tiempo. Esta decisión está motivada por varias razones, tanto en lo que respecta al modelo como a sus implicaciones algorítmicas. Dejamos esta discusión para la Sección 5.4.4, con el fin de reflejar mejor la motivación detrás de esta decisión.

Notación	Valor	Descripción
g	9.81 metros por segundo cuadrado	Constante gravitacional
ρ	1.42 kilogramos por metro cúbico	Densidad del aire
cr	0.006	Coeficiente de resistencia a la rodadura
cd	0.9	Coeficiente de arrastre aerodinámico
FS	2.40 metros cuadrados	Superficie frontal
m_c	900 kilogramos	Masa en vacío

Tab. 5.2.1: Definiciones de variables y valores sugeridos según lo propuesto por Demir et al. [6]. Masa del vehículo y superficie del camión de reparto urbano Alke ATX 320E.

5.2.2. Modelo de consumo de batería

Consideramos el modelo propuesto en Goeke et al. [15], donde el consumo de batería depende de diferentes variables agrupadas en tres categorías: masa del vehículo, velocidad y condiciones del terreno. Sea $P(v, q)$ la *función de consumo instantáneo* de un vehículo que viaja a velocidad v con carga q, definida por la siguiente ecuación

$$P(v,q) = \left(\frac{1}{2} \cdot cd \cdot \rho \cdot FS \cdot v^2 + (m_c + m_u q) \cdot g \cdot (\sin(\alpha^G) + cr \cdot \cos(\alpha^G))\right) \cdot v. \tag{5.1}$$

El resto de los parámetros se definen en la Tabla 5.2.1. Nótese que la masa m_u de cada una de las q unidades cargadas en el vehículo no está especificada. Esta variable depende del tipo de mercancías a entregar, que depende del problema. De manera similar, la variable α^G indica la pendiente del terreno, que depende de las condiciones de la carretera.

Dado que nos enfocamos en los efectos de la congestión, adaptamos esta fórmula para lograr un compromiso razonable entre la simplicidad y la expresividad del modelo. Asumimos que la masa de cada unidad es despreciable en comparación con la masa del vehículo ($m_u = 0$), y que la pendiente es cero ($\alpha^G = 0$). Así, definimos $h_1 = \frac{1}{2} \cdot cd \cdot \rho \cdot FS$ y $h_2 = m_c \cdot g \cdot cr$, y reescribimos $P(v, q)$ de la siguiente manera:

$$P(v) = h_1 v^3 + h_2 v. \tag{5.2}$$

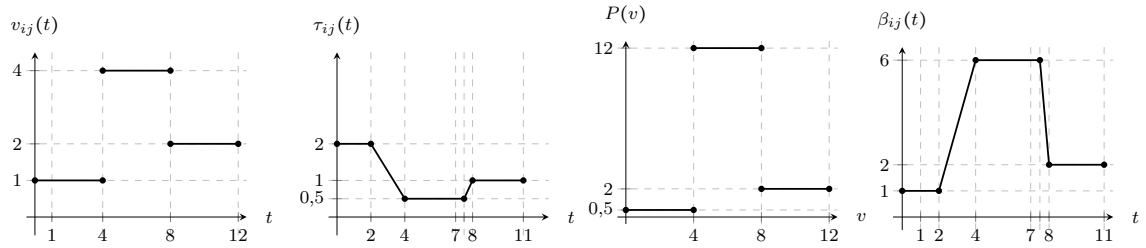
La Tabla 5.2.1 muestra los valores estimados para cada parámetro sugerido por Demir et al. [6]. Reemplazar estos valores en la anterior expresión resulta en $h_1 \approx 1{,}54$ y $h_2 \approx 52{,}97$. Así, podemos inferir que, bajo estas condiciones, $h_2 \approx 35h_1$. Esta relación se vuelve relevante para crear instancias sintéticas: dado un único índice de descarga h (por ejemplo, como se considera en el EVRPTW), se puede derivar una función de consumo instantáneo estableciendo $h_1 = \frac{h}{35}$ y $h_2 = h - \frac{h}{35}$, asumiendo $v = 1$ para el EVRPTW. Establecer una conexión análoga entre la velocidad promedio de viaje y las funciones de velocidad nos permitirá posteriormente evaluar los efectos de la congestión en comparación con el EVRPTW.

Observar que la función de consumo instantáneo $P(v)$ indica la energía consumida en un instante de tiempo por un vehículo que viaja a velocidad v. Sin embargo, la función de descarga de batería $\beta_{ij}(t)$ definida en la Sección 5.2.1 se refiere a la energía total consumida durante el recorrido del arco $(i, j) \in A$ si se parte desde i en el tiempo t. Para calcular la función de descarga de batería $\beta_{ij}(t)$ debemos combinar los tiempos de viaje dependientes del tiempo con las funciones de consumo instantáneo $P(v)$. Si un vehículo parte del vértice i hacia el vértice j en el tiempo t, entonces llega a j en el tiempo $t + \tau_{ij}(t)$. Así, el vehículo está recorriendo el arco (i, j) durante el intervalo $[t, t + \tau_{ij}(t)]$ y la descarga de batería $\beta_{ij}(t)$ puede calcularse integrando la función de consumo instantáneo sobre ese intervalo de tiempo.

$$\beta_{ij}(t) = \int_t^{t+\tau_{ij}(t)} P(v_{ij}(x))dx \tag{5.3}$$

Recordar que la función de velocidad $v_{ij}(t)$ sobre el arco $(i, j) \in A$ es una función escalonada; sea $|v_{ij}|$ su número de piezas. Si la k-ésima pieza de v_{ij} ($k = 1, \ldots, |v_{ij}|$) está definida en el dominio $[T_k, T_{k+1})$ y tiene un valor constante $\bar{v}_{ij}^k$, entonces la Ecuación (5.3) puede reescribirse como

$$\beta_{ij}(t) = \sum_{k=1}^{|v_{ij}|} P(\bar{v}_{ij}^k) \times |[T_k, T_{k+1}) \cap [t, t + \tau_{ij}(t)]| \tag{5.4}$$



(a) *(b)* *(c)* *(d)*

Fig. 5.2.1: Ejemplo numérico que ilustra la función de velocidad $v_{ij}(t)$ (5.2.1a); función de tiempo de viaje $\tau_{ij}(t)$ (5.2.1b); función de consumo instantáneo $P(v)$ (5.2.1c); y función de descarga de batería $\beta_{ij}(t)$ (5.2.1d) para un arco $(i,j) \in A$ con $d_{ij} = 2$.

De forma similar al caso de las funciones de tiempo de viaje, las funciones de descarga de batería son funciones lineales a trozos y continuas, como se establece en la siguiente proposición.

Observación 5.1. *Dado un arco $(i,j) \in A$, la función de descarga $\beta_{ij}(t)$ es lineal a trozos y continua.*

Demostración. El resultado se sigue ya que (5.4) es una composición de funciones lineales a trozos continuas. □

Para caracterizar completamente las funciones de descarga de la batería, es necesario describir sus puntos de quiebre. Cabe destacar que τ_{ij} se calcula de manera similar a β_{ij}, y por lo tanto estos puntos de quiebre son los mismos para τ_{ij} y β_{ij} Ichoua et al. [18].

La Figura 5.2.1 presenta un ejemplo de las diferentes funciones involucradas en el modelo de descarga de batería. Considere un arco $(i,j) \in A$ con una distancia $d_{ij} = 2$ y una función de velocidad v_{ij} como se especifica en la Figura 5.2.1a. Además, la flota de EVs tiene baterías con una función de consumo instantáneo $P(v) = \frac{1}{6}v^3 + \frac{1}{3}v$ (Figura 5.2.1c). Entonces, la función resultante de tiempo de viaje τ_{ij} y la función de descarga de batería β_{ij} se ilustran en las Figuras 5.2.1b y 5.2.1d, respectivamente.

Finalmente, recordemos la discusión de la Sección 5.1.1 respecto a las diferencias entre Goeke et al. [15] y Fetene et al. [12] en términos del consumo de energía y su dependencia de la velocidad de viaje. Aunque nos concentramos en el modelo propuesto por Goeke et al. [15], destacamos que nuestro enfoque es lo suficientemente general como para incorporar otras funciones de descarga definiendo la función de consumo instantáneo apropiada de las velocidades medias de viaje.

5.2.3. Tiempos de espera en la partida de los clientes

Frecuentemente, los planes tácticos asumen que los vehículos salen del cliente inmediatamente después de que el servicio se completa. Este es el caso en la mayoría de las variantes clásicas de los TDVRP, por sus siglas en inglés, motivado por el hecho de que los tiempos de viaje satisfacen la propiedad FIFO y, por lo tanto, demorar la partida no puede mejorar la solución.

Del modelo presentado en la Sección 5.2.2, los EV requieren un enfoque diferente. La velocidad de viaje (y, por lo tanto, el tiempo de viaje) y el nivel de batería suelen estar inversamente correlacionados, lo que significa que una ruta factible (eventualmente, la óptima) puede beneficiarse de viajar a una velocidad más lenta, cumpliendo aún con todas las restricciones operativas, para aumentar la autonomía del EV. Para ilustrar esta situación, recordemos el ejemplo de la Figura 5.2.1. Si el vehículo parte de i a $t_i = 7$, entonces el tiempo de viaje es $\tau_{ij}(7) = 0{,}5$, y llega a j a las 7.5 con un consumo de batería de $\beta_{ij}(7) = 6$ unidades de energía. Sin embargo, si es factible, demorar la salida hasta $t'_i = 8$ incrementa el tiempo de viaje a $\tau_{ij}(8) = 1$, llegando a j a las 9, pero consumiendo menos energía ya que $\beta_{ij}(8) = 2$. Notar que esta reducción en el consumo de energía puede ser crítica para las visitas a otros clientes, y por lo tanto necesaria para la solución

óptima. Entonces, el modelo debe permitir que un vehículo demore su salida desde una ubicación para garantizar la optimalidad, convirtiéndose también en una variable de decisión y aumentando la dificultad. A continuación, ofrecemos una interpretación para esta característica y discutimos una posible implementación práctica.

Para evitar soluciones que incluyan tiempos de espera poco realistas, limitamos la demora en la partida a $b_i + s_i$ al visitar $i \in V$ (eventualmente, hasta T para una estación $i \in V_s$). Reconocemos que, incluso con esta limitación, estos ajustes pueden ser difíciles de implementar en la práctica. Sin embargo, existen algunas interpretaciones interesantes desde un punto de vista gerencial. Primero, pueden considerarse como un buffer ante posibles demoras inesperadas durante las operaciones. Segundo, estos tiempos de espera sugieren que la velocidad podría reducirse al ejecutar la ruta. Para ilustrar esta segunda idea, recordar el ejemplo anterior y supongamos que el vehículo está listo para partir de i en $t_i = 7$, pero la solución óptima indica demorar la partida hasta $t'_i = 8$. Entonces, en lugar de demorar la partida, el vehículo podría salir aún a las $t_i = 7$ pero apuntar a llegar a j a las 9, como sería el caso al partir a $t'_i = 8$. Simplificando el ejemplo, el viaje promedio de i a j a una velocidad promedio de viaje $\hat{v}_{ij} = 1$ en la práctica, sin violar ninguna restricción de tiempo. Como se mencionó antes, asumimos en este caso que velocidades más lentas consumen menos batería.

Tener en cuenta que esto no se determina en la etapa de planificación al resolver el TDEVRPTW ya que las velocidades de viaje y los patrones de congestión no son una variable de decisión, sino una entrada para el problema. Esta decisión está parcialmente motivada por el hecho de que, en la práctica, de manera similar a otros VRPs, los ajustes en ruta suelen aplicarse en tiempo real debido a desviaciones del horario planificado. Para los TDVRPs en general, y el TDEVRPTW en particular, esto es especialmente importante ya que la congestión es un fenómeno dinámico que puede sufrir variaciones significativas respecto a los patrones estándar debido a eventos inesperados, más allá de la congestión clásica de la hora pico. Entonces, los ajustes en la velocidad de viaje originados por una partida demorada podrían implementarse durante la ejecución en lugar de determinarse en la etapa de planificación, en tiempo real, sugiriendo al conductor un rango deseado (eventualmente, óptimo) para la velocidad media de viaje en un enlace dado. De esta manera, el rango de velocidad sugerido puede tener en cuenta no sólo la estimación de congestión actualizada y más detallada, sino también la información específica sobre el contexto operativo actual, como el tiempo y las posibles demoras, entre otros.

Desde un punto de vista algorítmico, sugerir un rango óptimo de velocidad de viaje requiere resolver un problema adicional de optimización en tiempo real. Fukasawa et al. [14] abordan un problema conjunto de ruteo de vehículos y optimización de velocidad (JVRSP, por sus siglas en inglés) para el VRP considerando ICEVs tradicionales. Si se adaptara para el ruteo con EVs, este problema integrado podría constituir el bloque fundamental de una herramienta de soporte a la decisión en tiempo real para complementar el TDEVRPTW. Primero, observamos que el JVRSP es un problema muy desafiante, tanto teórica como prácticamente. Segundo, la re-optimización de una ruta debe incorporar todas las restricciones operativas del TDEVRPTW. Esto incluye el modelo de descarga de batería, eventualmente con información más detallada y actualizada sobre la velocidad de viaje. Además, las ventanas de tiempo deben seguir siendo satisfechas, imponiendo implícitamente un límite inferior para la reducción de la velocidad. Si el modelo de consumo de energía está alineado con el patrón observado en Fetene et al. [12], las sugerencias deberían incorporar, ya sea implícita o explícitamente, una velocidad mínima recomendada de viaje, ya que viajar demasiado lento puede no ser beneficioso ni para las ventanas de tiempo ni para la autonomía de conducción. Cabe destacar que la re-optimización también puede requerir aumentar la velocidad de viaje si la ejecución actual está retrasada para cumplir con las ventanas de tiempo, introduciendo eventualmente recargas adicionales, si es necesario. Finalmente, se señala que este marco puede generalizarse aún más para manejar otros tipos de eventos, llegando eventualmente a convertirse en una herramienta de gestión de interrupciones en tiempo real.

5.2.4. Consumo de batería y rutas factibles.

Tener en cuenta que el nivel de batería, incluyendo tanto el consumo como las recargas, no afecta el costo de una ruta (y, por lo tanto, la solución global) ya que solo restringe su factibilidad. Entonces, dado un camino p y un tiempo t, nos enfocamos en las combinaciones de decisiones que conducen al nivel máximo de batería cuando p está listo en el tiempo t, descartando otras soluciones dominadas. A continuación, desarrollamos el modelo considerando una política de recarga parcial, aunque puede adaptarse fácilmente a otros contextos. De manera similar a las variantes clásicas independientes del tiempo, asumimos que el tiempo de servicio s_i para el vértice i ya está codificado en la función de tiempo de viaje $\tau_{ij}(t)$. Los detalles específicos se proporcionan en la Sección 5.3. Primero consideramos una definición simplificada para un arco o vértice dado, y luego generalizamos la idea para caminos.

Supongamos que se nos da una función continua y lineal a trozos λ tal que $\lambda(t)$ indica el *nivel máximo de batería* de un vehículo que está listo para partir de un vértice i en un tiempo t. Además, supongamos que el dominio de λ es un intervalo cerrado $\text{dom}(\lambda)$. Al viajar por un arco $(i,j) \in A$ y partir de i en el tiempo t, la batería consumida viene dada por $\beta_{ij}(t)$ y, por lo tanto, el nivel de batería al llegar a j es $\lambda(t) - \beta_{ij}(t)$. Sin embargo, este no tiene que ser el nivel máximo de batería alcanzable para visitar j en el tiempo $t + \tau_{ij}(t)$ cuando se viaja directamente desde i. En efecto, el vehículo puede haber llegado a j en un tiempo t_{early} anterior a $t + \tau_{ij}(t)$, con más carga en la batería, y haber esperado hasta ese tiempo (lo cual no afecta el nivel de batería). Definimos la función $TRV_{ij}^{\lambda}(t)$ para capturar este comportamiento. Es decir, $TRV_{ij}^{\lambda}(t)$ denota el *nivel máximo de batería* al llegar a j en un tiempo t tras recorrer el arco $(i,j) \in A$, dado que la función de nivel máximo de batería en i es λ. Entonces, $TRV_{ij}^{\lambda}(t)$ se define como

$$TRV_{ij}^{\lambda}(t) = \max_{t' \in \text{dom}(\lambda) | \max\{a_j, t' + \tau_{ij}(t')} \{\lambda(t') - \beta_{ij}(t')\}. \tag{5.5}$$

Recordar que las funciones λ, β_{ij} y τ_{ij} son continuas y lineales a trozos, y que τ_{ij} además satisface la propiedad FIFO. Por lo tanto, $\lambda - \beta_{ij}$ y $t \to \max\{a_j, t + \tau_{ij}(t)\}$ también son continuas y lineales a trozos. Para simplificar, abusamos de la notación y definimos un dominio restringido considerando tiempos de llegada t en j tales que el nivel máximo de batería es no negativo ($TRV_{ij}^{\lambda}(t) \geq 0$), que sigue siendo un intervalo cerrado. Esta definición elimina los tiempos de llegada infactibles en j que no pueden alcanzarse porque la batería está agotada. Notar que $TRV_{ij}^{\lambda}(t)$ codifica implícitamente los tiempos de espera incurridos al demorar eventualmente la partida desde j, ya que considera tiempos de finalización factibles anteriores a t.

Un análisis similar se aplica a las paradas en las estaciones de recarga. En este caso, además asumimos que las funciones de recarga $g_i(w_0, t)$ son continuas y lineales a trozos, y proporcionamos en la Sección 5.3 la motivación para esta hipótesis. Una vez más, supongamos que se nos da una función continua y lineal a trozos μ tal que $\mu(t)$ indica el nivel máximo de batería de un vehículo que está listo para comenzar una recarga en el tiempo t en una estación $i \in V_s$. Nuevamente, μ está definida sobre un intervalo cerrado $\text{dom}(\mu)$ y asumimos que la imagen de μ está incluida en $\text{dom}(g_i)$. Sea $CHG_i^{\mu}(t)$ el *nivel máximo de batería* cuando la recarga se completa en el tiempo t, definido como

$$CHG_i^{\mu}(t) = \max_{t' \leq t, t' \in \text{dom}(\mu)} \{\mu(t') + g_i(\mu(t'), t - t')\}. \tag{5.6}$$

Nuevamente, dado que diferentes niveles de batería son factibles para un t dado, $CHG_i^{\mu}(t)$ indica el máximo. Como la recarga no tiene costo, siempre es conveniente cargar en lugar de esperar (sin recargar). De manera similar al caso anterior, la función $CHG_i^{\mu}(t)$ también captura posibles demoras incurridas por tiempos de espera, mientras que el dominio $\text{dom}(CHG_i^{\mu})$ de CHG_i^{μ} es igual al intervalo cerrado $\text{dom}(\mu)$.

Tanto TRV_{ij}^{λ} como CHG_i^{μ} se definen en términos de funciones lineales a trozos, aunque su estructura es un poco más compleja que una composición directa.
La Proposición 5.3 definida en el Apéndice 5.A.1 es relevante desde un punto de vista práctico, ya que demuestra que son funciones lineales a trozos y permite calcular TRV_{ij}^{λ} y CHG_i^{μ} considerando

solo un subconjunto finito de puntos de quiebre. La función G en dicha proposición generaliza la estructura de TRV_{ij}^{λ} y CHG_i^{μ}. Esto es sencillo de verificar para TRV_{ij}^{λ} al considerar $f(t',t) := \lambda(t') - \beta_{ij}(t')$ y $b(t) := \max\{t' | \max(a_j, t' + \tau_{ij}(t')) \leq t\}$. Lo mismo puede hacerse para CHG_i^{μ}, reemplazando $f(t',t) := \mu(t') + g_i(\mu(t'), t - t')$ y $b(t) := t, b(x) := \min dom(\mu)$).

Ahora extendemos estas ideas a caminos, con el fin de calcular el nivel de batería en cualquier momento dado de un viaje. Sea $p = (o = v_1, v_2, \ldots, v_k)$ un camino que comienza en el depósito o, pero que no necesariamente termina en d. Definimos $\lambda_{v_i}^p(t)$ como el nivel máximo de batería en un tiempo de finalización t al visitar el vértice v_i a lo largo del camino p, para $i = 1, \ldots, k$. Recordar que t es un tiempo de finalización cuando se puede partir de v_i en el tiempo t, lo que significa que v_i ya ha sido "procesado". Ciertamente, cada $t \in [0, T]$ es un tiempo de finalización para el depósito v_1. Por lo tanto, dado que cada vehículo parte de v_1 con un nivel de batería completo, se sigue que $\lambda_{v_1}^p(t) = B$ para todo $t \in [0, T]$. Claramente, $\lambda_{v_1}^p$ es una función continua y lineal a trozos definida sobre un intervalo cerrado. Para definir $\lambda_{v_i}^p$ procedemos por inducción sobre i, aprovechando las funciones TRV y CHG. Sea $\lambda = \lambda_{v_{i-1}}^p$, es decir, $\lambda(t)$ es el nivel máximo de batería cuando el vértice v_{i-1} está listo en el tiempo t. Si v_i es un cliente, entonces cada tiempo de llegada es un tiempo de finalización, por lo que $\lambda_{v_i}^p(t) = TRV_{v_{i-1}v_i}^{\lambda}(t)$. Recordar que $\lambda_{v_i}^p$ es una función continua y lineal a trozos definida en un intervalo cerrado, tal como lo requiere la inducción. Si v_i es una estación de recarga, entonces solo se puede partir de v_i después de que la batería esté recargada. Por lo tanto, debemos considerar el tiempo requerido para viajar de v_{i-1} a v_i más el tiempo para recargar en v_i. En consecuencia, $\lambda_{v_i}^p(t) = CHG_{v_i}^{\mu}(t)$ donde μ es la función tal que $\mu(t) = TRV_{v_{i-1}v_i}^{\lambda}(t)$. Una vez más, $\lambda_{v_i}^p(t)$ es una función continua y lineal a trozos definida en un intervalo cerrado.

Las funciones $\lambda_{v_i}^p$ se utilizan para codificar el nivel de batería dentro de nuestro algoritmo de labeling, como se discute en la Sección 5.4.4. Además, proporcionamos un ejemplo numérico que ilustra estas definiciones en 5.A.2.

5.3. Un framework general para tiempos dependientes del tiempo

A continuación describimos cómo integrar de manera natural al TDEVRPTW aspectos adicionales relacionados con la gestión del tiempo, previamente estudiados en la literatura de forma aislada, sin cambiar la estructura del problema. De hecho, todas estas características adicionales pueden codificarse directamente como una instancia del TDEVRPTW, tal como se describió en la sección anterior.

5.3.1. Manejo de los tiempos de servicio

De manera similar a los problemas independientes del tiempo, el tiempo de servicio s_i de un cliente $i \in V_c$ puede ser codificado directamente en las funciones de tiempo de viaje y descarga de batería para simplificar el modelo y las definiciones. Las siguientes ecuaciones describen cómo transformar una instancia $\mathcal{I}$ del TDEVRPTW en una instancia equivalente $\hat{\mathcal{I}}$ donde se puede asumir que los tiempos de servicio son cero. Sea τ_{ij} y β_{ij} las funciones de tiempo de viaje y descarga de batería para el arco (i, j) en $\mathcal{I}$, y $\hat{\tau}_{ij}$ y $\hat{\beta}_{ij}$ las funciones correspondientes en $\hat{\mathcal{I}}$. Entonces, para $\hat{\mathcal{I}}$ se define

$$\hat{\tau}_{ij}(t) := s_i + \tau_{ij}(t + s_i) \tag{5.7}$$

$$\hat{\beta}_{ij}(t) := \beta_{ij}(t + s_i). \tag{5.8}$$

y $s_i = 0$ para $i \in V$. Esta transformación asume que el depósito final s_d no tiene tiempo de servicio ($s_d = 0$). Además, notar que las funciones de tiempo de viaje siguen cumpliendo la propiedad FIFO. Para evitar confusiones con las fórmulas anteriores, en el resto del documento asumimos que el tiempo de servicio ya está codificado dentro de las funciones, pero mantenemos la notación original de τ_{ij} para los tiempos de viaje y β_{ij} para la descarga de batería.

5.3.2. Tiempos de carga no lineales

Se han considerado diferentes enfoques en la literatura relacionada para el modelo de recarga de baterías. El EVRPTW, en su configuración original, considera que el tiempo de recarga es lineal respecto a la cantidad de energía a recargar, independientemente de si la batería está vacía o, por ejemplo, medio cargada. Según Montoya et al. [27], este tiempo de recarga no es lineal. Además, estiman que recargar el 75 % inicial de la batería requiere un tiempo similar al final 25 %. Para cada estación, $j \in V_s$, sea $\hat{g}_j(t)$ la energía (en kWh) recargada en t unidades de tiempo en un vehículo con la batería vacía. Demuestran que considerar $\hat{g}_j(t)$ como una función lineal a trozos cóncava continua con tres tramos resulta en una buena estimación de los tiempos de recarga basada en datos recopilados. Además, esta función es inversible.

Usando estas estimaciones, podemos modelar el proceso de recarga de manera más realista. Intuitivamente, la tasa a la que un vehículo eléctrico se carga depende de cuán llena está la batería: cuando la batería está casi vacía, la carga es rápida, pero a medida que se acerca a estar llena, la tasa disminuye. Esto significa que la energía ganada en un tiempo dado no es simplemente proporcional al tiempo dedicado a la recarga, especialmente cuando la batería comienza parcialmente cargada.

Para capturar este comportamiento, definimos la función $g_j(w_0, t)$, que representa la energía ganada en la estación j después de cargar durante t unidades de tiempo comenzando desde un nivel de batería w_0. Esto se calcula como

$$g_j(w_0, t) = \min\left\{B - w_0,\ \hat{g}_j\left(\hat{g}_j^{-1}(w_0) + t\right) - w_0\right\}, \tag{5.9}$$

donde $\hat{g}_j(t)$ da la energía cargada en la estación j en t unidades de tiempo para una batería vacía, y B es la capacidad de la batería.

En otras palabras, primero determinamos cuánto tiempo habría tomado alcanzar el nivel de batería actual w_0 partiendo desde vacío. Luego añadimos el tiempo adicional de carga t, calculamos la energía total que se cargaría en ese tiempo, y restamos el nivel inicial w_0 para encontrar la ganancia neta. Esta función es continua y lineal a trozos por construcción, tal como se requiere.

5.3.3. Tiempos de espera con dependencia temporal

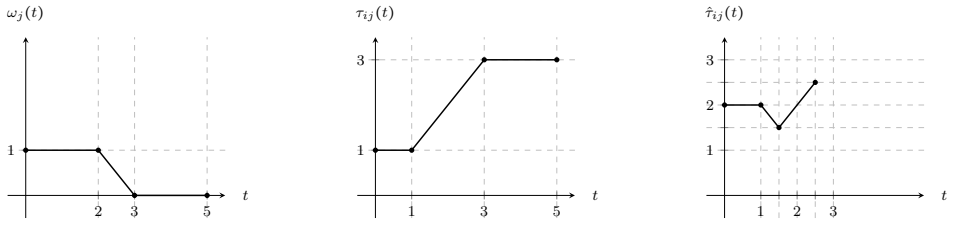
Como sugieren Keskin et al. [22], la suposición de que un vehículo puede realizar una recarga al llegar a una estación de carga a veces puede conducir a soluciones poco realistas. Esta observación es relevante en el contexto de la logística de media distancia y urbana, ya que es probable que se produzcan congestiones en las estaciones de recarga debido al aumento de la demanda durante las horas pico. Para capturar este comportamiento, ellos introducen una función de tiempo de espera con dependencia temporal $\omega_s(t)$ para cada estación $s \in V_s$ que indica el tiempo de espera esperado hasta que un terminal esté disponible si un vehículo llega a la estación s en el tiempo t. Estas funciones son continuas, lineales a trozos y satisfacen la propiedad FIFO.

Los tiempos de viaje con dependencia temporal incorporan naturalmente estos tiempos de espera siguiendo un procedimiento similar al descrito para los tiempos de servicio. Para mayor simplicidad, supongamos que $\omega_i(t) = 0$ para todo $t \in [0, T]$ cuando $i \in V \setminus V_s$. De manera similar a la Sección 5.3.1, dado un vértice $j \in V$, entonces la función de tiempo de viaje para el arco $(i, j) \in A$ se redefine como

$$\hat{\tau}_{ij}(t) := \tau_{ij}(t) + \omega_j(t + \tau_{ij}(t)). \tag{5.10}$$

Notar que esta transformación asume que no hay distinción entre los tiempos de viaje y de espera respecto a la función objetivo. De lo contrario, las funciones de tiempo de espera podrían requerir ser manejadas explícitamente. Nuevamente, para evitar confusiones, asumimos que los tiempos de espera con dependencia temporal están codificados directamente en la función τ_{ij}.

Una propiedad clave del modelo de tiempos de viaje con dependencia temporal radica en la condición FIFO. La siguiente proposición establece que la propiedad se mantiene después de aplicar esta transformación. Se omite la demostración ya que es la consecuencia directa de sumar y componer funciones lineales a trozos continuas que satisfacen la condición FIFO.



(a) *(b)* *(c)*

Fig. 5.3.1: Ejemplo de los tiempos de espera con dependencia temporal preprocesados incluyendo la función de tiempo de espera $\omega_j(t)$ de $j \in V_s$ (Figura 5.3.1a); $\tau_{ij}(t)$ del arco $(i,j) \in A$ (Figura 5.3.1b); $\tau_{ij}(t)$ del arco (i,j) después de (5.10) (Figura 5.3.1c)

Proposición 5.1. *La propiedad FIFO se mantiene para las funciones de tiempo de viaje después de aplicar (5.10).*

Sin embargo, obsérvese que los puntos de quiebre que definen τ_{ij} y β_{ij} pueden no permanecer iguales después de aplicar este preprocesamiento, como se indica en la Sección 5.2.2.

Proporcionamos un ejemplo numérico para ilustrar esta transformación en la Figura 5.3.1. Sea $j \in V_s$ una estación que tiene una función de tiempo de espera ω_j (Figura 5.3.1a). Considere un arco $(i,j) \in A$, y sea τ_{ij} su función de tiempo de viaje (Figura 5.3.1b). La función preprocesada τ_{ij} se muestra en la Figura 5.3.1c.

Finalmente, destacamos que se puede desarrollar un análisis similar para modelar tiempos de servicio con dependencia temporal mediante funciones lineales a trozos continuas. Aunque no consideramos explícitamente este escenario, pueden incorporarse de manera similar a nuestro marco de trabajo. Esto puede volverse relevante en la práctica, por ejemplo, para modelar tiempos variables de descarga debido a dificultades para encontrar espacios de estacionamiento durante las horas pico o casos de uso similares que surgen con frecuencia en logística.

5.3.4. Características adicionales de la infraestructura

Finalmente, cabe señalar que algunos EV admiten múltiples modos de carga, como diferentes velocidades o tecnologías de carga para un rango dado de capacidad de batería. Esto puede modelarse definiendo una función de carga específica para cada caso. Brevemente, cada estación está asociada con un *modo de carga* específico, que indica la función de recarga disponible. Para estaciones de recarga que tienen múltiples modos, simplemente podemos replicar la estación. Un costo fijo para cada modo también puede adicionalmente incorporarse en los arcos entrantes a cada estación.

5.4. Algoritmo exacto

Esta sección describe los componentes principales del algoritmo exacto BCP. Explicamos cómo adaptar el framework para alternar entre las diferentes políticas de recarga e incluimos una discusión sobre funciones objetivo alternativas. Destacamos que nuestro algoritmo extiende y generaliza el propuesto por Desaulniers et al. [7] y constituye un framework unificador para varios problemas interesantes relacionados con la planificación con vehículos eléctricos.

5.4.1. Preprocesamiento

Reducción de arcos y ventanas de tiempo

Se aplican dos pasos interrelacionados que buscan arcos infactibles y reducen las ventanas de tiempo, respectivamente. Estos pasos se repiten mientras haya cambios en la estructura de la

instancia. Para este procedimiento, seguimos las ideas de Lera-Romero et al. [24] y definidas en la Sección 3.5.1 que extienden las reglas clásicas propuestas por Desrosiers et al. [9]. Brevemente, un arco $(i,j) \in A$ se elimina si es infactible por capacidad, es decir, $q_i + q_j > Q$, o infactible por tiempo, es decir, $a_i + \tau_{ij}(a_i) > b_j$. Además, consideramos como entrada para el TDEVRPTW las instancias TDVRPTW preprocesadas, con las ventanas de tiempo ya ajustadas de acuerdo con los tiempos de viaje dependientes del tiempo. Omitimos los detalles por brevedad, pero remitimos a Desrosiers et al. [9] para una descripción detallada.

Batería mínima requerida

A continuación introducimos una regla de preprocesamiento diseñada específicamente para el TDEVRPTW. Para cada vértice $i \in V$ podemos calcular una cota inferior $MBR(i)$ del *mínimo nivel de batería requerido* para alcanzar cualquier estación de recarga (o eventualmente el depósito) desde el vértice i sin agotar la batería. Sea $\underline{\beta}_{ij} = \min\{\beta_{ij}(t) \mid t \in \text{dom}(\beta_{ij})\}$ una cota inferior del consumo de energía para el arco $(i,j) \in A$. Entonces, para $i \in V$, una cota inferior $MBR(i)$ puede calcularse mediante un algoritmo estándar de camino más corto usando $\underline{\beta}_{ij}$ como pesos de arco. Notar que $MBR(i) = 0$ para $i \in V \setminus V_c$. Además, los arcos $(i,j) \in A$ que satisfacen $B - \underline{\beta}_{ij} < MBR(j)$ indican que el nivel de batería al llegar a j no puede ser suficiente para alcanzar otra estación o el depósito, y por lo tanto pueden descartarse con seguridad. Estas cotas se utilizan además para mejorar las reglas de factibilidad del algoritmo de labeling y así reducir el tamaño del árbol de enumeración, como se describe en la Sección 5.4.4.

5.4.2. Formulación de *set partitioning*

Los algoritmos BCP y las formulaciones extendidas basadas en el modelo de *set paritioning* se consideran uno de los enfoques más efectivos para abordar diferentes variantes del VRP. Sea Ω el conjunto de todas las rutas factibles para el TDEVRPTW. Para cada ruta $r \in \Omega$, c_r representa su costo y la constante a_{ir} indica si la ruta r visita al cliente $i \in V$. Sea y_r una variable binaria que indica si una ruta $r \in \Omega$ es seleccionada en la solución óptima. La formulación de set partitioning para el TDEVRPTW se define de la siguiente manera.

$$\min \sum_{r \in \Omega} c_r y_r \tag{5.11}$$

$$\text{s.t.} \sum_{r \in \Omega} a_{ir} y_r = 1, i \in V_c \tag{5.12}$$

$$y_r \in \{0,1\}, r \in \Omega. \tag{5.13}$$

La función objetivo (5.11) minimiza el costo total de la solución, mientras que las restricciones (5.12) garantizan que cada cliente sea visitado exactamente una vez y (5.13) imponen el dominio de las variables.

Dado que Ω es exponencial, el enfoque clásico es resolver la relajación LP en cada nodo utilizando generación de columnas. Tal como se describe en la Sección 2.2, se inicializa un problema maestro restringido (RMP) con un subconjunto de las rutas $\Omega' \subseteq \Omega$. Para el TDEVRPTW, la inicialización del RMP es un poco más compleja en comparación con otras variantes, ya que la estrategia tradicional de seleccionar las rutas $r = \{o, i, d\}$ para definir Ω' puede ser infactible debido al consumo de batería. Para superar este problema, inicialmente incluimos en Ω' una ruta artificial (infactible) r que visita a todos los clientes con un costo $c_r = n \times \max_{(i,j) \in A} c_{ij}$ para garantizar la factibilidad de la relajación LP del RMP.

Después del paso de inicialización, se agregan nuevas columnas de forma iterativa hasta que el algoritmo converge a la solución óptima (fraccionaria). En cada iteración, se calcula la relajación LP del RMP para obtener las variables duales π_i asociadas con las restricciones (5.12) para $i \in V_c$. Usando esta información, si existe una ruta factible r con costo reducido negativo $\bar{c}_r = c_r - \sum_{i \in r} \pi_i$, entonces se agrega al RMP y se repite el procedimiento. De lo contrario, la solución fraccionaria actual es óptima. Para el TDEVRPTW, el problema de pricing se convierte en un

(a) (b)

Fig. 5.4.1: Ejemplo para variables de arco x_{ij} (ABR, 5.4.1a) y variables de cliente (CBR, 5.4.1b). La instancia tiene depósitos 0 y 9, estaciones de recarga $\{1, 2, 3\}$ y clientes $\{4, \cdots, 8\}$ donde se muestra una solución fraccionaria factible compuesta por cuatro rutas $r_1 = (0, 5, 3, 6, 1, 7, 8, 9)$, $r_2 = (0, 5, 3, 2, 4, 2, 9)$, $r_3 = (0, 3, 6, 1, 2, 4, 2, 9)$, $r_4 = (0, 2, 4, 2, 1, 7, 8, 9)$, con valores de variable $y_1 = \frac{2}{3}$ y $y_2 = y_3 = y_4 = \frac{1}{3}$. En el ABR, todos los arcos incidentes a un cliente tienen valores enteros, sin embargo la solución es fraccionaria.

Problema de Camino Mínimo Elemental Eléctrico con Dependencia Temporal y Restricciones de Recursos (TDEESPPRC, por sus siglas en inglés). Los enfoques exactos más efectivos para abordar problemas similares se basan en programación dinámica y algoritmos de labeling (Desaulniers et al. [7]). En general, se utilizan heurísticas para acelerar el desempeño del paso de pricing, y se ejecutan algoritmos exactos cuando no se encuentran rutas con costo reducido negativo para garantizar la optimalidad (ver Sección 5.4.4).

5.4.3. Esquema de branching

En esta sección, revisamos las ideas estándar consideradas para el EVRPTW y proponemos una regla de branching adaptada para gestionar las paradas intermedias en las estaciones de recarga. Los algoritmos clásicos de BCP consideran el branching sobre las llamadas *variables de arco*. Dada una solución óptima y^* de la relajación LP para (5.11)–(5.13), se define para cada arco $(i, j) \in A$ una variable binaria $x^*_{ij} = \sum_{r \in \Omega, (i,j) \in r} y^*_r$. Cuando no son posibles paradas intermedias entre clientes, como en el VRP, se puede demostrar fácilmente que y^* es fraccionario si y sólo si x^* también es fraccionario. Basándose en este resultado, se puede implementar una regla de branching *robusta* (véase, por ejemplo, Costa et al. [3]). Nos referimos a esta regla como la Regla de Branching por Arco (ABR, por sus siglas en inglés).

La formulación (5.11)–(5.13) codifica la factibilidad de una ruta, incluyendo las visitas a las estaciones de recarga, como parte de la definición de Ω. Nótese que las restricciones (5.12) se imponen únicamente para los vértices clientes $i \in V_c$, y de hecho Ω puede restringirse para considerar a lo sumo una ruta de costo mínimo para cada conjunto de vértices clientes $S \subseteq V_c$, ya que todas las demás rutas que visitan a los mismos clientes quedan dominadas. A partir de ahora, asumimos que Ω satisface esta propiedad ya que puede implementarse fácilmente y es una hipótesis clave para nuestro branching.

Basándonos en esta observación, proponemos una regla de branching alternativa que aprovecha la secuencia de clientes visitados por una ruta, a la que denominamos Regla de Branching por Cliente (Customer Branching Rule, CBR, por sus siglas en inglés). Sea $r = (0 = v_1, \cdots, v_k = d) \in \Omega$ una ruta factible. Definimos la *secuencia de clientes de* r, $CR(r)$, como la secuencia de vértices clientes en r, eliminando aquellos vértices $v_i \in V_s$. Dado $i, j \in V_c$ y $r \in \Omega$, sea b_{ijr} una constante que indica si j es visitado inmediatamente después de i en $CR(r)$. Intuitivamente, $b_{ijr} = 1$ si no hay clientes entre i y j en la ruta r. Definimos además variables binarias z_{ij} que indican si el vértice j es visitado inmediatamente después de i en alguna ruta $r \in \Omega$ sin considerar estaciones de recarga, las cuales pueden obtenerse a partir de la solución primal y como sigue

$$z_{ij} = \sum_{r \in \Omega} b_{ijr} y_r. \tag{5.14}$$

Dada una variable fraccionaria z_{ij}, creamos dos ramas descendentes, una para $z_{ij} = 0$ y otra para

$z_{ij} = 1$. La siguiente proposición demuestra que CBR es una regla de branching válida (ver 5.A.3 para la demostración).

Proposición 5.2. *Sea $(\bar{y}, \bar{z})$ una solución de la formulación (5.11)–(5.13) extendida con las variables z de (5.14). Entonces, $\bar{y}$ es una solución factible para el TDEVRPTW si y sólo si $\bar{z}_{ij} \in \{0, 1\}$ para todo $i, j \in V_c$.*

Simultáneamente con nuestro trabajo, Bessi et al. [1] discuten brevemente el uso del CBR. Nosotros además proporcionamos detalles específicos sobre nuestra implementación del CBR en nuestro marco de trabajo. Conceptualmente, el CBR puede manejarse de manera similar al ABR. La restricción $z_{ij} = 1$ puede ser reemplazada por la unión de las restricciones $z_{ik} = 0$ para $k \in V_c \setminus \{j\}$, y $z_{kj} = 0$ para $k \in V_c \setminus \{i\}$. Esto es particularmente útil al adaptar el problema de pricing para manejar esta estrategia de branching como se explica en la Sección 5.4.4. Finalmente, la Figura 5.4.1b muestra las variables z_{ij} asociadas a la solución presentada para la Figura 5.4.1a.

Otras alternativas de branching

Generalmente, el ABR se combina con algunas reglas adicionales para reducir aún más el árbol de enumeración. Para el EVRPTW, Desaulniers et al. [7] proponen una estrategia de branching de cuatro niveles. Primero, se realiza branching sobre el número de vehículos $\sum_{r \in \Omega} y_r$. Luego, el branching se realiza sobre el número total de recargas $\sum_{r \in \Omega} \rho_r y_r$, donde ρ_r indica el número de recargas realizadas en el recorrido de la ruta r. La siguiente decisión se basa en el número de recargas en una estación dada $\sum_{r \in \Omega} \rho_{rj} y_r$, donde ρ_{rj} indica el número de recargas de la ruta r en la estación $j \in V_f$. Finalmente, se considera ABR por variables x_{ij} si ninguna de las anteriores es fraccionaria. Cabe destacar que estas reglas también pueden combinarse con el CBR.

5.4.4. El problema de pricing

Esta sección describe el algoritmo de labeling desarrollado para abordar el problema de pricing de la Sección 5.4.2 y las variantes de la Sección 5.3. También ofrecemos una breve discusión sobre el impacto de considerar funciones objetivo alternativas.

El algoritmo de labeling

Desarrollamos un algoritmo de forward labeling que explora todos los caminos factibles. Siguiendo la idea de la Sección 2.7, estos algoritmos generan implícitamente un árbol de enumeración donde cada nodo se denomina label y representa un camino parcial que comienza desde el depósito inicial o. Con el objetivo de superar el crecimiento exponencial, por lo general se incorporan reglas de poda para reducir el número de labels enumerados. En un contexto independiente del tiempo, cada label suele ser descartado o procesado. Sin embargo, los problemas con dependencia temporal requieren una consideración adicional, ya que los labels codifican algunos de sus valores de recursos como funciones sobre un dominio dado. Por lo tanto, las reglas parciales de poda eventualmente descartan algunos intervalos específicos en el dominio (véase, por ejemplo, Lera-Romero et al. [25]).

A continuación, presentamos el algoritmo de labeling Forward para el TDEVRPTW con una política de recarga parcial y recargas ilimitadas. Cada camino $p = (o, \cdots, v)$ está representado por un label $L = (prev, v, S, q, c, \lambda_v^p)$ donde $prev$ es un puntero al label padre, v es el último vértice, S es un conjunto de clientes inalcanzables, q es la demanda total, c es el costo, y λ_v^p es la función de nivel máximo de batería definida en la Sección 5.2.4, es decir, $\lambda_v^p(t)$ es el nivel máximo de batería en el tiempo de finalización t después de recorrer p. Para mayor simplicidad, sea $prev(L)$, $v(L)$, $S(L)$, $q(L)$, $c(L)$ y λ_L los componentes $prev$, v, S, q, c y λ_p^v del label L, respectivamente. Además, sea $p(L)$ el camino representado por L y sea $\text{dom}(L) = \text{dom}(\lambda_L) = [a(L), b(L)]$ los tiempos de finalización factibles en $v(L)$ después de recorrer p, con $a(L) = \text{mín}(\text{dom}(L))$ siendo el tiempo de llegada más temprano al vértice $v(L)$. Cuando $p(L)$ es una ruta, llamamos a L un *route label*.

Sea $\bar{c}_{ij} = c_{ij} - \pi_i$ los costos reducidos de los arcos incluyendo las variables duales del RMP. Dado que los vehículos salen del depósito con la batería llena, el label inicial está representado

por la tupla $(\perp, o, \{o\}, 0, 0, 0, \lambda_o)$, donde $\lambda_o(t) = B$ para $t \in [a_o, b_o]$. La siguiente regla indica cómo extender un label durante el proceso de enumeración.

Regla 5.1 (Extensión). *Dado un label L, la extensión a través de un arco $(v(L), w)$ resulta en un nuevo label L_w donde $prev(L_w) = L$, $v(L_w) = w$, $S(L_w) = S(L) \cup \{w\}$, $q(L_w) = q(L) + q_w$, $c(L_w) = c(L) + \bar{c}_{vw}$, y la función λ_{L_w} se calcula a partir de λ_L como se discute en la Sección 5.2.4, es decir,*

$$\lambda_{L_w} = \begin{cases} \lambda_w^{p(L_w)}(t) = TRV^{\lambda}_{v(L)w}(t) & \text{si } w \in V_c, \\ \lambda_w^{p(L_w)}(t) = CHG^{\mu}_w(t), \text{ con } \mu(t) = TRV^{\lambda}_{v(L)w}(t) & \text{si } w \in V_s. \end{cases}$$

Notar que L_w es factible cuando $w \notin S(L)$, $q(L_w) \leq Q$ y $a(L) + \tau_{vw}(a(L)) \leq b_w$.

Los requisitos de batería permiten la introducción de reglas de factibilidad adicionales y específicas. Recordar de la Sección 5.4.1 que dado un vértice $v \in V$, $MBR(v)$ indica la batería mínima requerida para partir desde un vértice v y alcanzar una estación o, alternativamente, el depósito final sin agotar la batería. La siguiente regla aprovecha esta información para descartar algunos intervalos de $\text{dom}(L)$.

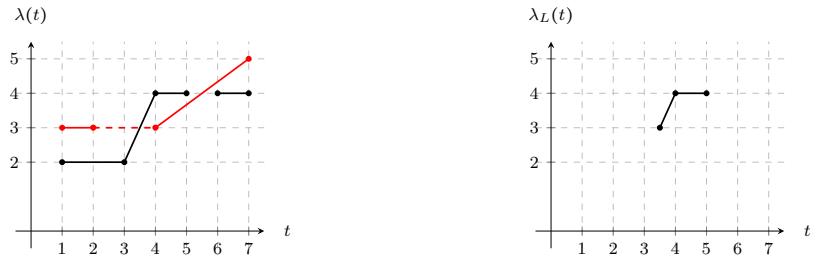
Regla 5.2 (Factibilidad). *Dado un label L, cualquier tiempo $t \in dom(L)$ tal que $\lambda_L(t) < MBR(v(L))$ puede descartarse de manera segura de λ_L. Si $dom(\lambda_L) = \varnothing$ entonces $p(L)$ es* infactible *y L puede descartarse.*

Dado que la función λ_L es no decreciente por definición, cualquier intervalo descartado de $\text{dom}(\lambda_L)$ debe ser un prefijo. Este hecho puede reducir el tiempo requerido para calcular esta regla en muchos labels.

Otra acción para reducir el número de labels enumerados se basa en aplicar reglas de dominancia. Brevemente, dado dos labels L, M, si toda extensión factible p' de $p(L)$ también es una extensión factible de $p(M)$ pero $p(M) + p'$ siempre tiene un costo menor, entonces L puede ser descartado. En la práctica, para reducir la carga computacional, se prueban condiciones suficientes para asegurar que un label pueda ser descartado. En un contexto con dependencia temporal, la dominancia parcial permite descartar algunos instantes de tiempo específicos dominados del dominio $\text{dom}(L)$, incluso si el label no está completamente dominado. Como resultado, el dominio de un label puede convertirse en una colección de intervalos en lugar de uno solo. Sin embargo, esto puede mitigarse asumiendo que el vehículo simplemente espera sin consumir batería en este intervalo dominado, lo cual se modela considerando una función constante que extiende la función desde el trozo no dominado previo (remitimos a Lera-Romero et al. [25] para más detalles). Para evitar introducir definiciones adicionales, abusamos de la notación y asumimos que las funciones de nivel de batería λ_L y λ_M, así como sus dominios correspondientes, se ajustan de esta manera en la siguiente regla.

Regla 5.3 (Dominancia parcial). *Sean L y M dos labels que satisfacen (i) $v(L) = v(M)$, (ii) $q(M) \leq q(L)$, (iii) $c(M) \leq c(L)$, y (iv) $S(M) \subseteq S(L)$. Entonces, cualquier tiempo $t \in dom(\lambda_L) \cap dom(\lambda_M)$ tal que (v) $\lambda_L(t) \leq \lambda_M(t)$ puede ser descartado de $dom(L)$. Si $dom(\lambda_L) = \varnothing$ entonces decimos que L está* completamente dominado *y puede ser descartado de forma segura.*

Desde el punto de vista de la implementación, para acelerar las pruebas de dominancia consideramos la estructura clásica de bucket donde una tabla de programación dinámica almacena labels indexados por sus recursos v, q. Dentro de cada bucket, los labels están ordenados por costo. Observar que las pruebas de dominancia incluyen una comparación de las funciones de nivel de batería entre dos labels, lo cual requiere tiempo lineal respecto al número total de piezas en ambas funciones. Por brevedad, omitimos los detalles de la implementación. La Figura 5.4.2 muestra un ejemplo del algoritmo de dominancia parcial cuando se verifica la dominancia del label L frente al label M, donde la línea discontinua representa la extensión constante sobre un intervalo dominado en el dominio del label M. Esta extensión constante es válida porque, al llegar antes a un nodo, el vehículo siempre puede esperar sin incurrir en consumo adicional de batería. Por lo tanto, la función de nivel de batería puede extenderse con una pieza constante sobre el intervalo dominado. La Figura 5.4.2b muestra las piezas restantes de L después de la dominancia.



(a) (b)

Fig. 5.4.2: Ejemplo de la dominancia parcial aplicada a L (negro) y M (rojo) (5.4.2a) y de los fragmentos restantes (5.4.2b)

Incorporación de CBR al algoritmo de pricing

La regla de branching CBR impuesta sobre las variables z_{ij} puede ser gestionada dentro del algoritmo de pricing de manera similar al branching sobre las variables clásicas de arco x_{ij}, previniendo que tales rutas sean generadas en lugar de añadir explícitamente las restricciones $z_{ij} = 0$.

En cada nodo del árbol de branch and bound, se define $F_i = \{j \in V_c \cup \{d\} \mid z_{ij} = 0\}$ como el conjunto de *sucesores prohibidos* para cada cliente $i \in V_c \cup \{o, d\}$ (y se toma $F_i = \varnothing$ para cada $i \in V_s$). Intuitivamente, F_i codifica el conjunto de sucesores de i que no pueden ser visitados debido a decisiones de branching tomadas anteriormente en el árbol. Incluimos en la representación de un label L un recurso adicional u que indica el último vértice de $V_c \cup \{o, d\}$ visitado en $p(L)$, es decir, ya sea un cliente o el depósito. La extensión de L a través de un arco $(v(L), w) \in A$ debe además verificar que $w \notin F_{u(L)}$. En consecuencia, la Regla 5.3 debe modificarse adecuadamente para seguir siendo válida, donde también debe cumplirse que $F_{u(M)} \subseteq F_{u(L)}$ para concluir que el label M domina al label L en el tiempo t. Esta nueva condición asegura que toda extensión factible de L no involucra a $w \in F_{u(M)}$ como el siguiente vértice, y por lo tanto también es una extensión factible de M. Fortalecemos aún más esta condición considerando el conjunto de clientes ya visitados $S(L)$ de la siguiente manera

$$F_{u(M)} \subseteq F_{u(L)} \cup S(L). \tag{5.15}$$

Incorporar la condición (5.15) a la Regla 5.3 afecta la eficiencia del algoritmo de labeling. De manera similar al ABR para VRPs sin paradas intermedias entre clientes, se descartan extensiones de label porque pueden volverse infactibles. Por el contrario, las reglas de dominancia se vuelven más débiles debido a la introducción de una nueva condición. Claramente, esto implica que se enumeran más labels, aunque destacamos que el efecto de la condición (5.15) se limita únicamente a aquellos labels L que terminan en una estación de recarga, de lo contrario $v(L) = v(M) = u(L) = u(M)$ y la inclusión se satisface trivialmente.

Políticas alternativas de recarga

Hasta ahora se ha presentado el modelo de consumo de batería y el algoritmo de labeling bajo una política de recarga parcial. Afortunadamente, adaptar el modelo para considerar una política de recarga completa es sencillo. La Figura 5.A.1e ilustra la función del nivel de batería después de aplicar el paso de carga. Notar que existe un prefijo del dominio $P = [\text{mín}(\text{dom}(\lambda_j^p)), t_2] \subseteq \text{dom}(\lambda_j^p)$ donde $\lambda_j^p(t) < B = 10$ para todo $t \in P$. Bajo una política de recarga completa, es infactible partir desde una estación de recarga $j \in V_s$ con ese nivel de batería.

Formalmente, sea p un camino que termina en una estación $j \in V_s$, y sea λ_j^p su función de nivel de batería en el vértice j. Entonces, todos los instantes de tiempo $t \in \text{dom}(\lambda_j^p)$ tales que $\lambda_j^p(t) < B$ deben ser descartados. Nótese que una política de recarga completa puede imponerse estableciendo $MBR(j) = B$ para todas las estaciones de recarga $j \in V_s$, ya que la Regla 5.2 elimina cualquier tiempo del dominio por debajo de este valor.

Otra variante introducida por Desaulniers et al. [7] es la posibilidad de restringir el número de recargas en ruta. Dado un número máximo de recargas por ruta R_{max}, se extiende la definición de un label L para considerar un nuevo recurso $r(L)$ que indique el número de recargas realizadas, y se prohibe una extensión a una estación de recarga si $r(L) \geq R_{max}$. Adicionalmente, la regla de dominancia 5.3 debe modificarse incluyendo la condición $r(M) \leq r(L)$, que garantiza que todas las extensiones de L también sean factibles para M.

Heurísticas de pricing

La ejecución de la generación de columnas generalmente se acelera recurriendo a heurísticas para identificar columnas con un costo reducido negativo, mientras que el algoritmo exacto de labeling se ejecuta si no se identifica ninguna columna de este tipo. Un tipo de adaptación heurística frecuentemente utilizada para los algoritmos de labeling es reducir el tamaño del grafo. Al inicio de cada iteración de pricing, el grafo se reduce para considerar solo los k arcos con dependencia temporal salientes $(i, j) \in A$ desde cada vértice $i \in V$ con el menor costo reducido $\bar{c}_{ij}$. Nos referimos a este enfoque como la heurística k-shrink.

Consideramos además relajar la Regla de dominancia 5.3 dentro del algoritmo de pricing, manteniendo la factibilidad de las soluciones pero eventualmente descartando rutas con un costo reducido negativo debido a un criterio de dominancia más débil. Nos referimos a la heurística donde se relaja la condición (iv) de la Regla 5.3 como *relax-S*, mientras que la heurística que relaja la condición (v) (es decir, el nivel de batería) se denomina *relax-B*.

Funciones objetivo alternativas

Aunque el TDEVRPTW minimiza el costo total de las rutas, incluimos una breve discusión respecto a la factibilidad y el impacto de considerar funciones objetivo alternativas adicionales. En algunos problemas con dependencia temporal, como el TDTSP (véase, por ejemplo, Cordeau et al. [2]), el objetivo es minimizar el makespan de la ruta para un vehículo que parte al inicio del horizonte de planificación, es decir, en el tiempo $t_0 = 0$. Nuestro enfoque puede adaptarse fácilmente para considerar tal función objetivo. En este caso, el costo reducido de un label L en el tiempo $t \in \text{dom}(L)$ se define como $t - \sum_{i \in p(L)} \pi_i$, donde π son los valores de las variables duales. Dentro del algoritmo de pricing, fijar el costo $c_{ij} = 0$ para todos los arcos $(i, j) \in A$ redefine $c(L)$ como la suma de los precios duales del camino $p(L)$ para un label L. Notar que la función objetivo no afecta las reglas de extensión, las cuales permanecen sin cambios. Sin embargo, las reglas de dominancia dependen del costo reducido de la ruta. Para un label L, al alcanzar $v(L)$ en algún tiempo t el costo reducido es $t - c(L)$, y por lo tanto la condición (iii) de la Regla 5.3 compara efectivamente el costo reducido de dos labels L y M en un tiempo fijo t. Notar que incluso cuando la dominancia parcial puede descartar algunos intervalos específicos, la ruta óptima conserva su makespan en el dominio.

Se considera una función objetivo más compleja para el TDVRPTW (ver, por ejemplo, Dabia et al. [4] y Lera-Romero et al. [25]) donde se permite que los vehículos demoren su partida desde el depósito con el fin de minimizar la duración de la ruta. Desafortunadamente, manejar esta función objetivo dentro del algoritmo BCP propuesto para el TDEVRPTW introduce una complejidad significativa en comparación con el TDVRPTW. La razón es que se añade una nueva variable de decisión, que es el tiempo inicial de partida t_0. Dado un camino p, el nivel de batería λ_p depende tanto del tiempo de finalización t como del tiempo inicial de partida t_0, y se convierte en una función bidimensional debido a las demoras tanto en la partida desde el depósito como en la partida desde un vértice, como se discute en la Sección 5.2.3. Aunque el procedimiento de dominancia puede adaptarse, se volvería computacionalmente demasiado costoso almacenar este tipo de función dentro de cada label y realizar las comparaciones correspondientes en el paso de dominancia. De manera similar, considerar costos dependientes de la energía para el camino durante el viaje sufriría la misma desventaja. Sin embargo, cabe destacar que un costo fijo de recarga c_j puede incluirse fácilmente para cada recarga realizada en la estación $j \in V_s$ definiendo $c_{ij} := c_{ij} + c_j$ para $(i, j) \in A$.

5.4.5. Planos de corte

Incorporamos las *Subset Row Inequalities* (SRCs, por sus siglas en inglés) propuestas por Jepsen et al. [21] como planos de corte para fortalecer la relajación LP, con el objetivo de reducir el tamaño del árbol de branching. Estas desigualdades válidas se han convertido en un estándar dentro de los algoritmos exactos para VRP basados en la formulación de set partitioning (véase, por ejemplo, Costa et al. [3]) y también han sido utilizadas para el EVRPTW por Desaulniers et al. [7]. Formalmente, un SRC $s = (S, k)$ está definido por un conjunto de clientes $S \subseteq V_c$, y un coeficiente $k \in \mathbb{N}$. Dada una ruta $r \in \Omega$ y $s = (S, k)$, sea $v_{sr} = \left\lfloor \frac{|S \cap r|}{k} \right\rfloor$ el número de subconjuntos en S que tienen k vértices que son visitados por r. Entonces, el SRC se expresa como

$$\sum_{r \in \Omega} y_r v_{sr} \leq \left\lfloor \frac{|S|}{k} \right\rfloor. \tag{5.16}$$

Consideramos el *3-SRC*, es decir, con $|S| = 3$ y $k = 2$. Cabe señalar que, dado que los SRC son cortes no robustos, remitimos al lector a Costa et al. [3] para los detalles sobre cómo adaptar el problema de pricing. Brevemente, nuestro algoritmo de planos de corte es el siguiente. Después de resolver la relajación LP mediante generación de columnas, se añaden iterativamente a la formulación un máximo de n_{cuts} desigualdades violadas siguiendo un criterio de violación máxima, reoptimizando el RMP en cada paso sin resolver el problema de pricing. Experimentos preliminares indican que este enfoque tiende a reducir el número de cortes añadidos al RMP mientras se preserva su impacto en la calidad de la relajación LP.

5.5. Configuración de branch-and-price

Esta sección describe los componentes principales del algoritmo branch-and-price utilizado para resolver el problema.

Rutas añadidas por iteración En cada iteración de pricing, se generan y añaden un hasta 3000 rutas con costo reducido negativo al problema maestro. Para limitar la redundancia, sólo se conserva la mejor ruta para cada subconjunto único de clientes visitados. Como resultado, el número de rutas añadidas por iteración puede variar y no está acotado a priori. Esta estrategia equilibra la generación de columnas mejorantes diversas con la eficiencia computacional.

Estrategia de branching El esquema de branching en nuestro algoritmo se centra en los arcos (o secuencias de dos clientes, en el caso de CBR) con valor fraccionario en las variables x. Cuando se encuentra una solución fraccionaria, el algoritmo identifica un subconjunto prioritario cuyo valor está más cerca de 0.5 y aplica strong branching sobre estos candidatos, creando nodos hijos que prohíben o imponen el uso del arco (o secuencia de dos clientes) seleccionado. La lógica de branching se adapta según la estrategia elegida, asegurando que los nuevos nodos reflejen decisiones basadas en clientes o en arcos. Entre las opciones evaluadas, el algoritmo selecciona aquella con la mejora más prometedora en la cota y añade los nodos correspondientes al árbol de búsqueda.

5.6. Experimentos computacionales

En esta sección, evaluamos el modelo con dependencia temporal propuesto en la Sección 5.2 y el algoritmo BCP desarrollado en la Sección 5.4 para el TDEVRPTW. Para ello, presentamos un conjunto de instancias de referencia con dependencia temporal, generadas como una extensión de las propuestas por Desaulniers et al. [7], tal como se describe en la Sección 5.6.1. Los experimentos están diseñados para obtener insights de negocio sobre la importancia de incorporar tiempos de viaje con dependencia temporal y, además, proporcionar evidencia sólida acerca de la efectividad de nuestro enfoque mediante experimentos extensos con y sin velocidades de viaje variables. Los experimentos se realizan en un Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz con 32GB de RAM, y los algoritmos están desarrollados en C++ y utilizan CPLEX 12.9 como solver de LP.

5.6.1. Configuración experimental

Para evaluar los diferentes componentes del modelo, extendemos las instancias propuestas en Desaulniers et al. [7] para el EVRPTW incorporando velocidades de viaje con dependencia temporal, tiempos de recarga no lineales y tiempos de espera con dependencia temporal de la siguiente manera.

Con respecto a las velocidades de viaje con dependencia temporal, seguimos la estrategia propuesta por Dabia et al. [4]. El horizonte de planificación $[0, T = b_d]$ se divide en cinco zonas de velocidad de viaje $T_1 = [0, 0{,}2T], T_2 = [0{,}2T, 0{,}4T], T_3 = [0{,}4T, 0{,}6T], T_4 = [0{,}6T, 0{,}8T], T_5 = [0{,}8T, 1{,}0T]$. A cada arco $(i, j) \in A$ se le asigna aleatoriamente uno de tres perfiles de velocidad que representan diferentes niveles de congestión de tráfico: *alta*, *normal* y *baja*. La Figura 5.6.1a muestra la función de velocidad v_p asignada a cada perfil de velocidad. Para permitir comparaciones cualitativas significativas, cada función de velocidad v_p tiene una velocidad promedio de 1.0 durante el horizonte de planificación, que es la velocidad de viaje considerada en las instancias con dependencia temporal. El parámetro r requerido para definir la función de consumo instantáneo $P(v)$ se obtiene de las instancias originales.

Todas las instancias se extienden con funciones lineales a trozos de recarga siguiendo el patrón propuesto por Montoya et al. [27]. Recordar que las instancias originales de EVRPTW de Desaulniers et al. [7] asumen que la carga de la batería es proporcional a la cantidad de energía a recargar, es decir, la carga completa de la batería con capacidad B toma t_g unidades de tiempo, recargando B/t_g kWh por unidad. Definimos las funciones lineales a trozos de recarga donde una recarga completa de una batería con capacidad B también requiere t_g unidades de tiempo, aunque la recarga por unidad de tiempo no es uniforme a lo largo del intervalo (Figura 5.6.1b). Además, cada estación pertenece a uno de los siguientes modos: *rápido*, *medio* y *lento*. El modo medio se define según B y t_g, mientras que los modos lento y rápido son una versión escalada del modo medio, $2t_g$ y $0{,}5t_g$, respectivamente.

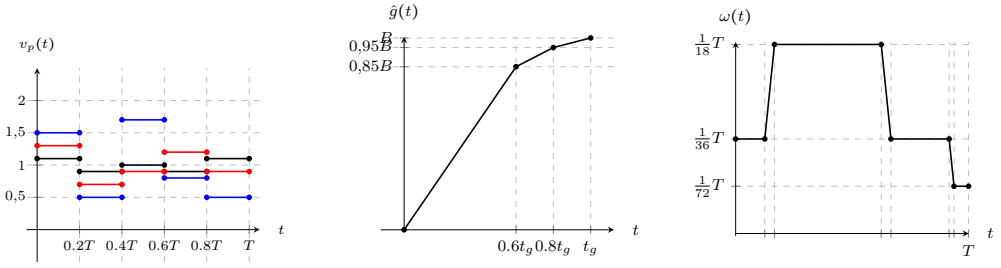
Con respecto a los tiempos de espera con dependencia temporal en las estaciones, Keskin et al. [22] consideran múltiples escenarios que capturan diferentes situaciones de la vida real. Como nuestro objetivo es más general que evaluar solo los tiempos de espera, las instancias se extienden de acuerdo con el patrón *TD-Smooth-Long*, ya que representa un escenario intermedio donde las estaciones no están ni demasiado congestionadas ni vacías. La Figura 5.6.1c muestra la función considerada, que se escala para cada instancia según el horizonte de planificación.

Tener en cuenta que las instancias de referencia propuestas por Desaulniers et al. [7] pueden recuperarse configurando todos los perfiles de velocidad a $v_p(t) = 1$, definiendo las funciones de recarga con un único tramo que requiere el mismo tiempo total que el cargador medio para cargar completamente la batería, y eliminando los tiempos de espera configurando $\omega_s(t) = 0$ para cada estación $s \in V_s$. Para los experimentos, definimos las siguientes configuraciones para las instancias:

- Básico: instancias de Desaulniers et al. [7] extendidas con información dependiente del tiempo.
- NC: Básico extendido con funciones de carga no lineales.
- WT: Básico extendido con funciones de tiempo de espera dependientes del tiempo.
- All: Básico extendido con funciones de carga no lineales y funciones de tiempo de espera dependientes del tiempo.

Para evaluar el impacto de las velocidades de viaje con dependencia temporal, consideramos además las versiones independientes del tiempo de los benchmarks mencionados anteriormente, como se indicó previamente.

La configuración de nuestro algoritmo BCP se basa en varios parámetros, algunos de ellos dependientes del tipo de instancias consideradas. Con respecto al pricing heurístico, ejecutamos relax-S, relax-B y k-shrink descritos en la Sección 5.4.4 con diferentes combinaciones de parámetros. Basándonos en experimentos preliminares limitados, la configuración es la siguiente: relax-S primero combinado con k-shrink para $k = 3, 7$ y 12; relax-B; relax-S; y finalmente 7-shrink. Una vez que una configuración falla, no se intenta de nuevo en iteraciones futuras, si las hay. Si todas las heurísticas fallan en encontrar una columna con costo reducido negativo, entonces se ejecuta el algoritmo exacto. En cuanto a los planos de corte, establecemos $n_{max} = 200$.



(a) *(b)* *(c)*

Fig. 5.6.1: Descripción del conjunto de datos, incluyendo la función de velocidad $v_p(t)$ para cada perfil p alto (azul), normal (rojo), bajo (negro) (Figura 5.6.1a); una función genérica de recarga $\hat{g}$ de Montoya et al. [27] (Figura 5.6.1b); y la función de tiempo de espera $\omega(t)$ de Keskin et al. [22] (Figura 5.6.1c)

La política de recarga considerada también impacta la configuración del algoritmo BCP, en particular, si se permiten recargas simples (S) o múltiples (M), y si están restringidas a ser completas (F) o parciales (P). Bajo las políticas de recarga SF y MF, establecemos $MBR(j) = B$ para todo $j \in V_s$. Para SF y SP, establecemos $R_{max} = 1$ para limitar el número de recargas en ruta a 1.

5.6.2. Impacto del TDEVRPTW

El modelo con dependencia temporal presentado en la Sección 5.2 incluye más información y, por lo tanto, se espera que produzca soluciones de mejor calidad. En esta sección, evaluamos y comparamos las soluciones obtenidas al considerar la dependencia temporal con sus contrapartes independientes del tiempo.

Para evaluar el modelo con dependencia temporal, inicialmente realizamos experimentos sobre los cuatro conjuntos de datos descritos en la sección anterior, con y sin velocidades de viaje dependientes del tiempo. Para los experimentos, consideramos la política MP, con múltiples paradas y recarga parcial, ya que representa el escenario más complejo y rico. Evaluamos la calidad de la solución en cada caso mediante cuatro métricas: costo total; la suma del makespan de las rutas; el número promedio de recargas por ruta (#rech); y el número de rutas (#routes). La Tabla 5.6.1 presenta los resultados, desagregados por conjunto de datos y promediados según el tamaño de instancia, únicamente sobre las instancias que se resuelven a optimalidad en ambos casos. La columna $|V_c|$ indica el número de clientes, la columna *común* muestra el número de instancias resueltas a optimalidad en ambos contextos. Para el caso independiente del tiempo, los promedios de cada métrica se reportan en términos absolutos y se usan como línea base. Sea z_{ti} y z_{td} el valor de una métrica dada para el caso independiente del tiempo y para el caso dependiente del tiempo, respectivamente. Para el escenario dependiente del tiempo, los valores se reportan como la diferencia relativa respecto al escenario independiente del tiempo, es decir, $(z_{td} - z_{ti})/z_{ti}$.

El mensaje principal de la Tabla 5.6.1 es que incorporar la congestión a nivel de planificación induce diferencias entre las soluciones óptimas con respecto a todas las métricas. Por ejemplo, el número de recargas por ruta aumenta en más de un 10 % en promedio y hasta un 24 % en el caso con dependencia temporal, lo que sugiere que simplificar las velocidades de viaje promediándolas impacta directamente en el modelo de descarga de batería. Sin embargo, observar que incluir la dependencia temporal no conduce necesariamente a soluciones de mayor costo, como es el caso de las instancias de 25 clientes para el conjunto de datos NC. De hecho, considerar la congestión explícitamente conduce a rutas más rápidas en varios casos, manteniéndose aún factibles. Además, destacamos las diferencias en el número de rutas, que pueden volverse significativas desde una perspectiva gerencial.

Por construcción, las instancias independientes del tiempo pueden interpretarse como el resultado de promediar las velocidades de viaje de las instancias con dependencia temporal. Para

			independiente del tiempo				dependiente del tiempo			
dataset	$\|V_c\|$	común	costo	makespan	#rech	#rutas	costo (%)	makespan (%)	#rech (%)	#rutas (%)
	25	56	458.50	2879.88	1.09	4.11	-0.20	-0.05	6.82	-3.04
Básico	50	43	702.85	4441.24	0.97	6.63	0.33	0.02	24.57	-0.70
	100	14	1251.98	5449.47	1.28	12.93	0.38	2.16	4.53	2.21
	25	56	457.03	2846.69	1.10	4.02	0.01	0.72	11.92	-2.22
NC	50	45	699.23	4256.58	1.08	6.51	0.55	0.84	14.13	0.00
	100	14	1188.15	6643.29	1.21	12.14	0.61	0.50	4.07	2.94
	25	56	462.79	2948.02	0.93	4.21	0.06	-0.01	5.31	-0.85
WT	50	43	723.78	4582.24	0.89	6.88	0.46	0.91	9.68	2.36
	100	13	1224.89	7364.81	0.90	13.54	0.65	0.89	7.74	-0.57
	25	56	461.17	2906.51	0.94	4.11	0.15	0.09	5.23	-0.87
TODOS	50	44	717.97	4480.35	0.88	6.91	0.69	0.58	13.98	-0.99
	100	12	1238.16	6825.26	0.90	13.17	0.48	1.75	11.35	3.16

Tab. 5.6.1: Comparación de la calidad de soluciones óptimas: independiente del tiempo vs dependiente del tiempo.

dataset	$\|V_c\|$	#inst	#inf-s	%inf	%inf-t	%inf-b	%Δ-t	%Δ-b
	25	56	47	50.22	30.49	23.32	1.47	0.76
Basic	50	48	41	60.07	40.28	26.15	1.61	0.54
	100	17	17	82.16	58.92	34.05	3.61	0.88
	25	56	47	50.45	33.64	23.64	1.49	0.61
NC	50	48	42	57.34	37.20	25.94	1.76	0.60
	100	16	16	74.29	48.57	32.00	3.71	0.76
	25	56	46	50.43	28.21	25.21	1.57	0.67
WT	50	48	42	62.05	39.93	28.38	1.68	0.49
	100	18	18	87.43	54.86	41.14	3.60	1.12
	25	56	48	50.44	27.19	25.88	1.54	0.61
ALL	50	48	42	57.14	36.88	27.57	1.66	0.57
	100	15	15	74.23	47.24	36.81	3.46	1.05

Tab. 5.6.2: Análisis de factibilidad de las soluciones independientes del tiempo evaluadas en un contexto dependiente del tiempo.

resaltar los efectos de la congestión, proponemos un análisis posterior evaluando las soluciones obtenidas para las instancias independientes del tiempo bajo el contexto dependiente del tiempo. En este sentido, asumimos que las instancias con dependencia temporal representan una mejor aproximación de las operaciones del mundo real, y las infactibilidades detectadas son el resultado de descuidar la dependencia temporal y sus efectos. Nos enfocamos en dos fuentes diferentes de infactibilidades. Decimos que una ruta r es *infactible por tiempo* si es infactible incluso cuando la capacidad de la batería no es restrictiva, es decir, $B = \infty$. Intuitivamente, una ruta infactible por tiempo es infactible ya sea por la violación de una ventana de tiempo o porque su makespan excede el horizonte de planificación. De manera similar, decimos que una ruta r es *infactible por batería* si se eliminan todos los tiempos de cierre de las ventanas de tiempo, es decir, fijando $b_i = T$ para todo $i \in V$, y r sigue siendo infactible debido a batería insuficiente. Estas infactibilidades pueden medirse y cuantificarse adicionalmente. Definimos la *violación de tiempo* como la peor violación de la ventana de tiempo de un cliente dividida por el tamaño del horizonte T, y la *violación de batería* como la energía extra requerida para hacer la ruta factible dividida por la capacidad de la batería B. Finalmente, consideramos la solución como infactible si al menos una de sus rutas es infactible.

La Tabla 5.6.2 presenta los resultados de las pruebas de infactibilidad, agregados por n, para cada conjunto de datos. La columna #inst indica el número de instancias resueltas de manera óptima en el contexto independiente del tiempo. El resto de las columnas indican los resultados de las métricas descritas anteriormente. La columna #inf-s representa el número de soluciones infactibles, y las columnas %inf, %inf-t y %inf-b representan el número de rutas infactibles, rutas infactibles por tiempo y rutas infactibles por batería, respectivamente, sobre el total de rutas. %Δ-t y %Δ-b indican la violación promedio de las rutas infactibles por tiempo y las rutas infactibles por batería, respectivamente.

De los resultados, observamos que un gran número de soluciones obtenidas a partir del modelo independiente del tiempo se vuelven infactibles cuando se evalúan bajo el modelo con dependencia temporal. Al inspeccionar las soluciones, notamos que en promedio más del 60 % de las rutas son infactibles. De manera similar al TDVRP clásico, un porcentaje considerable de estas infactibilidades se debe a una violación de las ventanas de tiempo originada en las variaciones en los tiempos de viaje, entre el 30 y 55 % de las rutas según nuestros resultados. Sin embargo, un resultado muy interesante es que un número significativo de infactibilidades se debe a que el EV se queda sin batería mientras ejecuta la ruta. Estos porcentajes oscilan entre el 20 y 40 %, dependiendo del escenario considerado. Desde un punto de vista gerencial, este tipo de infactibilidades puede resultar en costos considerablemente elevados. Finalmente, notamos que las violaciones promedio son bastante pequeñas en ambos casos. Esto puede estar relacionado con los parámetros considerados para las instancias. Perfiles de velocidad con mayor variabilidad y una relación diferente entre h_1 y h_2 podrían conducir a valores mayores.

Para comprender mejor el impacto práctico del modelo propuesto, ampliamos los experimentos anteriores considerando nuevos conjuntos de datos que incorporan algunos valores del mundo real, con el objetivo de incluir al menos parcialmente algunas características presentes en redes y flotas de vehículos en tiempo real. Los conjuntos de datos Basic, NC, WT y ALL descritos inicialmente se amplían en Basic-R, NC-R, WT-R y ALL-R, respectivamente, de la siguiente manera:

- Consideramos instancias con $n = 25, 50$ del conjunto de datos original.
- La información del tiempo de viaje (horizonte de planificación, velocidad de viaje, ventanas de tiempo, etc.) se escala para representar velocidades de viaje más realistas. En estos nuevos conjuntos de datos, la velocidad de viaje del EV varía aproximadamente entre 30 y 102 km/h, con un promedio de 60 km/h. El horizonte de planificación se expresa en horas (de 4 a 80 horas) y las distancias entre ubicaciones en km (entre 10 y 60 kilómetros).
- Algunos parámetros se ajustan siguiendo los reportados en Schiffer et al. [35]. Más específicamente, la capacidad de la batería y la autonomía obtenida al conducir a la velocidad promedio coinciden con las especificaciones de un Tesla Semi 500. La información respecto a clientes, ubicaciones y distancias no se utiliza ya que, hasta donde sabemos, no es pública.

Replicamos el experimento original utilizando este nuevo conjunto de instancias, enfocándonos en un análisis cualitativo. La Tabla 5.6.3 presenta los resultados obtenidos. La leyenda de la tabla es la misma que en la Tabla 5.6.1. Comparando estas dos tablas, observamos que incorporar la dependencia temporal en el modelo produce resultados similares. Las diferencias son consistentes en todas las métricas reportadas en términos relativos en ambos conjuntos de datos. El conjunto de datos escalado muestra un menor número de recargas requeridas por ruta, probablemente debido a que la capacidad de batería del Semi 500 es proporcionalmente mayor que la de las instancias anteriores. Como resultado, se requieren menos recargas por ruta y las diferencias entre conjuntos de datos son despreciables para $n = 25$, y más significativas para $n = 50$. En el primero, los tiempos de espera y los tiempos de carga no lineales tienen casi ningún impacto, ya que se realizan pocas recargas y las rutas tienen pocos vértices, minimizando el impacto potencial. Finalmente, cabe destacar que el makespan promedio reportado para las nuevas instancias es significativamente menor porque el tiempo se mide en horas.

La Tabla 5.6.4 replica el análisis cualitativo, utilizando las mismas referencias que la Tabla 5.6.2, mostrando algunos hallazgos interesantes. Cabe destacar que la dependencia temporal sigue siendo un factor importante respecto a la factibilidad, donde un número significativo de rutas obtenidas en un contexto independiente del tiempo se vuelven infactibles cuando se evalúan bajo el modelo con dependencia temporal. Sin embargo, el número de rutas infactibles debido a la agotamiento de la batería es relativamente moderado. Esto puede deberse a diferentes razones. Como se mencionó anteriormente, la capacidad de la batería es un recurso menos restrictivo en el nuevo conjunto de instancias. Además, la información del vehículo considera un contexto logístico de media distancia, donde el peso puede ser relevante. Adicionalmente, nuestras instancias aún asumen que el impacto

			independiente del tiempo				dependiente del tiempo			
dataset	$\|V_c\|$	común	costo	makespan	#rech	#rutas	costo (%)	makespan (%)	#rech (%)	#rutas (%)
Basic-R	25	56	450.59	46.43	0.35	3.84	-0.07	0.60	17.51	-1.40
	50	39	698.60	73.82	0.22	6.38	0.94	1.02	88.24	0.80
NC-R	25	55	450.72	46.65	0.37	3.89	-0.28	0.50	37.22	-2.80
	50	37	696.53	75.98	0.25	6.49	0.57	0.82	51.88	1.25
WT-R	25	56	450.59	46.49	0.26	3.89	0.13	0.57	-0.07	-1.38
	50	42	690.49	71.54	0.16	6.33	48.48	-0.12	49.00	-1.50
ALL-R	25	56	450.59	46.45	0.10	3.86	0.13	0.74	145.96	-75.46
	50	45	683.69	70.08	0.18	6.38	48.65	-0.26	36.87	-73.51

Tab. 5.6.3: Comparación de la calidad de soluciones óptimas: independiente del tiempo vs dependiente del tiempo, dataset escalado.

dataset	$\|V_c\|$	#inst	#inf-s	%inf	%inf-t	%inf-b	%Δ-t	%Δ-b
Basic-R	25	56	31	24.53	24.53	0.00	1.74	-
	50	41	25	26.69	26.69	0.40	2.28	0.91
NC-R	25	56	31	25.00	25.00	0.00	1.77	-
	50	40	24	25.93	25.93	0.41	2.30	0.94
WT-R	25	56	31	24.19	24.19	0.00	1.72	-
	50	43	25	25.57	25.57	0.38	2.18	0.87
ALL-R	25	56	31	24.07	24.07	0.00	1.71	-
	50	45	24	23.00	23.00	0.35	2.07	0.79

Tab. 5.6.4: Análisis de factibilidad de las soluciones independientes del tiempo evaluadas en un contexto con dependencia temporal, conjunto de datos escalado.

de la carga puede ser despreciado. También se observa que para instancias más grandes que requieren más recargas por ruta, aparece nuevamente un comportamiento similar al del conjunto de datos no escalado.

5.6.3. La regla de branching CBR

En esta sección, evaluamos el CBR dentro del framework propuesto. Para reducir el impacto potencial de factores externos, realizamos los experimentos sobre las instancias originales independientes del tiempo de Desaulniers et al. [7]. Además, tanto CBR como ABR se prueban en un BP puro (sin los SRCs) para apreciar mejor el impacto al enumerar más nodos en el árbol de branch and bound. Una observación importante es que el ABR puede implementarse eliminando arcos del dígrafo soporte ya que, para estas instancias, no existen arcos entre dos estaciones de recarga. Consideramos dicha implementación simplificada en este experimento, aunque destacamos que esto puede resultar beneficioso para el ABR en comparación con el caso general, dado que no se deben incluir restricciones adicionales en el RMP y no se eliminan aristas del grafo subyacente.

La Tabla 5.A.1 en 5.A.4 presenta los resultados detallados. La conclusión principal es que usar CBR en este contexto es beneficioso, ya que BP-CBR es capaz de cerrar 15 instancias más que BP-ABR. Una de las razones es el número de nodos enumerados, donde CBR obtiene reducciones de aproximadamente un 50 % en promedio. Este efecto es más evidente en instancias más grandes. Sin embargo, destacamos que la ejecución promedio disminuye a un ritmo menor. Recordar que el problema de pricing necesita ser modificado dentro del CBR, y algunas iteraciones de pricing pueden volverse más costosas en términos de los tiempos de cómputo debido a esta sobrecarga, mientras que el ABR se implementa eliminando arcos.

5.6.4. Performance del BCP

Realizamos extensos experimentos computacionales para evaluar el desempeño del algoritmo BCP bajo diferentes contextos y configuraciones. El primer experimento compara la efectividad

			independiente del tiempo								dependiente del tiempo							
dataset	$\|V_c\|$	inst	opt	time	nodes	cuts	rCols	tCols	%eg	%cg	opt	time	nodes	cuts	rCols	tCols	%eg	%cg
	25	56	56	7.05	1.04	2.61	2219.00	2220.43	0.50	0.00	56	25.13	1.14	2.70	2270.75	2274.55	0.43	0.01
Básico	50	56	48	193.27	2.88	14.79	11933.58	11999.83	0.80	0.06	44	197.67	2.00	15.23	11284.19	11471.16	0.79	0.09
	100	56	17	775.94	11.82	73.05	22141.85	24231.59	1.96	0.27	17	822.30	10.88	65.05	48783.71	58343.88	2.02	0.34
	25	56	56	8.72	1.25	2.29	2149.25	2158.34	0.46	0.02	56	43.16	1.11	3.25	2493.89	2498.39	0.49	0.02
NC	50	56	48	350.40	2.83	15.13	13961.58	14149.81	0.69	0.06	47	291.80	1.51	15.04	11045.96	11140.26	0.72	0.05
	100	56	16	536.48	9.63	63.00	52014.05	34475.38	1.91	0.22	15	446.46	7.67	58.74	36713.32	31578.73	1.64	0.19
	25	56	56	5.64	1.21	2.80	2106.36	2141.41	0.59	0.02	56	21.84	1.75	3.41	2162.05	2204.52	0.59	0.05
WT	50	56	48	219.54	6.75	16.75	10418.23	10703.42	0.95	0.10	44	165.87	3.45	15.65	9527.37	9757.80	0.90	0.12
	100	56	18	907.41	20.00	77.68	45534.16	21655.06	2.07	0.43	14	727.61	16.86	61.57	50785.70	18950.93	2.34	0.70
	25	56	56	6.77	1.21	2.13	2131.71	2134.91	0.43	0.01	56	41.76	1.04	2.54	2303.34	2303.38	0.48	0.00
TODOS	50	56	48	259.72	3.75	17.13	11543.60	11720.15	0.78	0.06	47	263.20	3.21	14.83	12255.94	12517.98	0.82	0.08
	100	56	15	530.57	14.87	69.04	52582.96	22963.67	2.49	0.76	13	489.25	14.54	69.76	37435.19	19116.54	2.34	0.54

Tab. 5.6.5: Rendimiento para el BCP para todas las variantes, con y sin dependencia temporal y bajo una política de recarga MP.

del BCP en las cuatro configuraciones (Basic, NC, WT y All) tanto para las versiones con dependencia temporal como sin ella, restringidas a la política de recarga MP para mantener manejable el número de experimentos. Nuevamente, seleccionamos esta política ya que representa el escenario más desafiante. La Tabla 5.6.5 reporta el número de instancias resueltas de forma óptima (opt), el tiempo promedio de ejecución (time), el promedio de nodos enumerados (nodes), el promedio de cortes añadidos en el nodo raíz (cuts) y el promedio de columnas añadidas al resolver el nodo raíz (rCols) y durante toda la ejecución (tCols). Para evaluar el impacto de los SRCs, reportamos el promedio del % gap (%eg) de la primera relajación LP y también en el nodo raíz (%cg). Nuevamente, estas métricas se reportan de forma agregada para cada conjunto de datos y $|V_c|$. Las columnas time, nodes y tCols se promedian sobre las instancias resueltas de forma óptima para cada métrica, mientras que las columnas cuts, rCols, %eg y %cg se promedian sobre las instancias donde el nodo raíz fue cerrado.

La Tabla 5.6.5 sugiere que la incorporación de las diferentes variantes tiene un impacto limitado en el desempeño de nuestro algoritmo. Este es el comportamiento esperado para nuestro framework con dependencia temporal, ya que los tiempos de espera dependientes del tiempo están codificados directamente en las funciones de tiempo de viaje y las funciones no lineales de carga se manejan en el paso de extensión del algoritmo de labeling. Sin embargo, destacamos que las instancias con dependencia temporal parecen ser ligeramente más difíciles de resolver en comparación con sus contrapartes independientes del tiempo, manteniendo todas las demás variables fijas. Por ejemplo, el número de columnas requeridas para ejecutar el algoritmo está en el mismo orden independientemente de la variante. El problema de pricing se vuelve más difícil de resolver en un contexto con dependencia temporal, lo cual es razonable considerando que la función del nivel de batería dentro de cada label tiene potencialmente más puntos de quiebre. De hecho, observamos que el BCP tiende a enumerar menos nodos en un escenario con dependencia temporal comparado con su contraparte independiente del tiempo para tiempos de ejecución comparables. Los planos de corte tienen un impacto positivo causando una reducción dramática del gap en el nodo raíz. Como se esperaba, el número total de cortes aumenta conforme aumenta n. Nosotros notamos que el BCP produce gaps notables en el nodo raíz, por debajo del 1 % en promedio, como resultado de combinar rutas elementales con SRCs.

5.6.5. Comparación con otros algoritmos exactos

Los algoritmos exactos basados en formulaciones extendidas superan a los modelos compactos en casi todos los contextos, convirtiéndose en el estándar actual dentro de la literatura sobre VRP. Este patrón es aún más pronunciado en el caso de los TDVRP, ya que la relajación LP de las formulaciones compactas tiende a ser más débil debido a la presencia de tiempos de viaje variables. Para proporcionar más evidencia sobre la efectividad de nuestro marco utilizando una línea base de alto nivel, establecemos una comparación directa entre nuestro BCP y los resultados reportados por Desaulniers et al. [7] sobre las instancias originales de EVRPTW con las cuatro políticas de recarga (SF, SP, MF, MP). Además del TDEVRPTW, este es un experimento interesante por sí

mismo ya que los algoritmos BCP incorporan diferentes componentes. En resumen, Desaulniers et al. [7] implementa un algoritmo de labeling bidireccional, relajaciones *ng*-route y un algoritmo de planos de corte diferente. Sin embargo, destacamos que nuestro marco incluye una sobrecarga significativa para gestionar funciones lineales a trozos.

Los resultados se muestran en la Tabla 5.A.2 en 5.A.4, donde la referencia es la misma que en los experimentos anteriores. En este caso, los promedios se calculan sobre todas las instancias resueltas a optimalidad por el método correspondiente. De hecho, nuestro algoritmo es capaz de resolver a optimalidad 13 y 45 instancias más en comparación con las versiones mono y bidireccionales de Desaulniers et al. [7], respectivamente. Esto indica que nuestro BCP es competitivo, aunque cabe destacar que nuestra computadora es aproximadamente un 20 % más rápida. Más allá de los tiempos de cómputo, las métricas restantes sugieren que algunos componentes nuevos del algoritmo tienen un impacto positivo. El número de nodos enumerados se mantiene moderado, particularmente para instancias con $n = 50, 100$. Los gap del nodo raíz comparten el mismo comportamiento, ya que son menores al 0.20 % en promedio. Creemos que esto se debe a una combinación de factores, a saber, considerar rutas elementales que conducen a cotas inferiores más ajustadas, el branching CBR y las reglas de preprocesamiento mejoradas. Destacamos que se reportan mejores resultados para el EVRPTW en Desaulniers et al. [8] utilizando un método mejorado de variable fixing.

5.7. Conclusiones

En este artículo, proponemos un modelo general para el TDEVRPTW que considera el efecto de la congestión en la descarga de la batería. Además, desarrollamos un marco general que unifica e integra varias características realistas considerando algunas de las restricciones operativas más relevantes para el EVRP de la literatura, hasta ahora abordadas de forma independiente en la literatura relacionada. Desde un punto de vista gerencial, mostramos que ignorar las velocidades de viaje con dependencia temporal a nivel de planificación puede afectar la calidad de los planes de ruteo, incurriendo potencialmente en violaciones de las ventanas de tiempo y excediendo el rango de conducción. Nuestros experimentos sugieren que hasta un 40 % de las infactibilidades del plan de distribución pueden ser causadas por exceder la capacidad de la batería de los EVs, y que este efecto es más moderado en escenarios menos estresados. Desde una perspectiva algorítmica, desarrollamos un nuevo algoritmo BCP que generaliza métodos existentes para el EVRPTW, capaz de manejar tiempos de espera con dependencia temporal, funciones de carga no lineales, tiempos de viaje con dependencia temporal y funciones de descarga de batería dependientes de la velocidad. Introducimos una nueva regla de branching derivada para gestionar eficientemente las paradas intermedias entre clientes. En general, el algoritmo propuesto demuestra ser muy efectivo en un amplio conjunto de instancias de un espectro diverso de escenarios, resolviendo instancias con hasta 100 clientes a optimalidad y con un desempeño comparable al de la literatura para el EVRPTW (independiente del tiempo).

Como investigación futura, creemos que la evaluación del modelo con datos reales es una de las direcciones más interesantes. Esto acercaría el modelo y los experimentos a las operaciones del mundo real. En términos del algoritmo, la incorporación de componentes adicionales como variable fixing podría aumentar el tamaño de las instancias resueltas. Un problema interesante y desafiante que busca reducir la brecha entre los modelos y su implementación práctica se refiere a diseñar algoritmos capaces de ajustar las rutas en tiempo real durante su ejecución para tener en cuenta posibles cambios en las condiciones del tráfico y otras desviaciones del horario original. Una línea de investigación prometedora se refiere a la adaptación del problema conjunto de ruteo de vehículos y optimización de velocidad estudiado en Fukasawa et al. [14] para considerar el ruteo con EVs, eventualmente considerando un marco heurístico y el vehículo único para re-optimizar cada ruta de forma independiente. Además, los modelos integrados que exploren el impacto tanto de la carga como de la velocidad serían un desafío algorítmico pero podrían conducir a conocimientos gerenciales significativos.

Referencias

[1] Dario Bessi et al. *A Route-Based Algorithm for the Electric Vehicle Routing Problem with Multiple Technologies*. Inf. téc. 2022.

[2] Jean-François Cordeau et al. «Analysis and Branch-and-Cut Algorithm for the Time-Dependent Travelling Salesman Problem». En: *Transport. Sci.* 48.1 (2014), págs. 46-58.

[3] Luciano Costa et al. «Exact Branch-Price-and-Cut Algorithms for Vehicle Routing». En: *Transport. Sci.* 53.4 (2019), págs. 946-985.

[4] Said Dabia et al. «Branch and Price for the Time-Dependent Vehicle Routing Problem with Time Windows». En: *Transportation Science* 47.3 (2013), págs. 380-396.

[5] Brian A. Davis et al. «A methodology to evaluate the competitiveness of electric delivery trucks». En: *Transport. Res. E* 49.1 (2013), págs. 8-23.

[6] Emrah Demir et al. «A review of recent research on green road freight transportation». En: *Eur. J. Oper. Res.* 237.3 (2014), págs. 775-793.

[7] Guy Desaulniers et al. «Exact algorithms for electric vehicle-routing problems with time windows». En: *Oper. Res.* 64.6 (2016), págs. 1388-1405.

[8] Guy Desaulniers et al. «Variable Fixing for Two-Arc Sequences in Branch-Price-and-Cut Algorithms on Path-Based Models». En: *Transp. Sci.* 54.5 (2020), págs. 1170-1188.

[9] Jacques Desrosiers et al. «Time constrained routing and scheduling». En: *Network Routing*. Vol. 8. Handbooks in Operations Research and Management Science. Elsevier, 1995, págs. 35-139.

[10] Sevgi Erdoğan et al. «A Green Vehicle Routing Problem». En: *Transport. Res. E* 109 (2012), págs. 100-114.

[11] Angel Felipe et al. «A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges». En: *Transport. Res. E* 71 (2014), págs. 111-128.

[12] Gebeyehu M Fetene et al. «Harnessing big data for estimating the energy consumption and driving range of electric vehicles». En: *Transportation Research Part D: Transport and Environment* 54 (2017), págs. 1-11.

[13] Aurélien Froger et al. «Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions». En: *Comput. Oper. Res.* 104 (2019), págs. 256-294.

[14] Ricardo Fukasawa et al. «A joint vehicle routing and speed optimization problem». En: *INFORMS J. Comput.* 30.4 (2018), págs. 694-709.

[15] Dominik Goeke et al. «Routing a mixed fleet of electric and conventional vehicles». En: *Eur. J. Oper. Res.* 245.1 (2015), págs. 81-99.

[16] H. Heni. «Optimization of time-dependent routing problems considering dynamic paths and fuel consumption». Tesis doct. Université Laval, 2018.

[17] Yixiao Huang et al. «Time-dependent vehicle routing problem with path flexibility». En: *Transp. Res. B* 95 (2017), págs. 169-195.

[18] Soumia Ichoua et al. «Vehicle dispatching with time-dependent travel times». En: *Eur. J. Oper. Res.* 144.2 (2003), págs. 379-396.

[19] Stefan Irnich. «Resource extension functions: properties, inversion, and generalization to segments». En: *OR Spectr.* 30.1 (2008), págs. 113-148.

[20] Stefan Irnich et al. «Shortest Path Problems with Resource Constraints». En: *Column Generation*. Ed. por Guy Desaulniers et al. Springer US, 2005, págs. 33-65.

[21] Mads Jepsen et al. «Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows». En: *Oper. Res.* 56 (2008), págs. 497-511.

[22] Merve Keskin et al. «Electric Vehicle Routing Problem with Time-Dependent Waiting Times at Recharging Stations». En: *Comput. Oper. Res.* 107 (2019), págs. 77-94.

[23] Merve Keskin et al. «Partial Recharge Strategies for the Electric Routing Problem with Time Windows». En: *Transport. Res. C* 65 (2016), págs. 111-127.

[24] Gonzalo Lera-Romero et al. «A branch and cut algorithm for the time-dependent profitable tour problem with resource constraints». En: *European Journal of Operational Research* 289.3 (2021), págs. 879-896. ISSN: 0377-2217.

[25] Gonzalo Lera-Romero et al. «Linear edge costs and labeling algorithms: The case of the time-dependent vehicle routing problem with time windows». En: *Networks* 76.1 (2020), págs. 24-53.

[26] Ji Lu et al. «The Time-Dependent Electric Vehicle Routing Problem: Model and Solution». En: *Expert Syst. Appl.* (2020), pág. 113593.

[27] Alejandro Montoya et al. «The electric vehicle routing problem with nonlinear charging function». En: *Transport. Res. B* 103 (2017), págs. 87-110.

[28] Samuel Pelletier et al. «Goods distribution with electric vehicles: Review and research perspectives». En: *Transport. Sci.* 50.1 (2016), págs. 3-22.

[29] Samuel Pelletier et al. «The electric vehicle routing problem with energy consumption uncertainty». En: *Transport. Res. B* 126 (2019), págs. 225-255.

[30] Sina Rastani et al. «A large neighborhood search-based matheuristic for the load-dependent electric vehicle routing problem with time windows». En: *Annals of Operations Research* (2021), págs. 1-33.

[31] J. Restrepo et al. «Performance testing of electric vehicles on operating conditions in Bogotá DC, Colombia». En: *2014 IEEE PES Transmission Distribution Conference and Exposition – Latin America*. 2014, págs. 1-8.

[32] R. Roberti et al. «The Electric Traveling Salesman Problem with Time Windows». En: *Transport. Res. E* 89 (2016), págs. 32-52.

[33] Ons Sassi et al. *Vehicle Routing Problem with Mixed fleet of conventional and heterogenous electric vehicles and time dependent charging costs*. Inf. téc. 2014.

[34] Martin W. P. Savelsbergh et al. «50th Anniversary Invited Article—City Logistics: Challenges and Opportunities». En: *Transport. Sci.* 50.2 (2016), págs. 579-590.

[35] Maximilian Schiffer et al. «Integrated planning for electric commercial vehicle fleets: A case study for retail mid-haul logistics networks». En: *European Journal of Operational Research* 291.3 (2021), págs. 944-960.

[36] Maximilian Schiffer et al. «The Electric Location Routing Problem with Time Windows and Partial Recharging». En: *Eur. J. Oper. Res.* 260 (2017).

[37] Maximilian Schiffer et al. «Vehicle routing and location routing with intermediate stops: A review». En: *Transport. Sci.* 53.2 (2019), págs. 319-343.

[38] Michael Schneider et al. «The electric vehicle-routing problem with time windows and recharging stations». En: *Transport. Sci.* 48.4 (2014), págs. 500-520.

[39] Sai Shao et al. «Electric Vehicle Routing Problem with Charging Time and Variable Travel Time». En: *Math. Probl. Eng.* 2017 (2017).

[40] Peng Sun et al. «The time-dependent capacitated profitable tour problem with time windows and precedence constraints». En: *Eur. J. Oper. Res.* 264.3 (2018), págs. 1058-1073.

[41] Mariam Tagmouti et al. «Arc routing problems with time-dependent service costs». En: *Eur. J. Oper. Res.* 181.1 (2007), págs. 30-39.

[42] Paolo Toth et al., eds. *Vehicle Routing: Problem, Methods and Applications*. Philadelphia, PA, USA: Society for Industrial y Applied Mathematics, 2014. ISBN: 0-89871-498-2.

[43] UPS. *Accelerating Sustainable Solutions, UPS 2019 Sustainability Progress Report.* 2019.

[44] Xinkai Wu et al. «Electric vehicles' energy consumption measurement and estimation». En: *Transport. Res. D* 34 (2015), págs. 52-67.

[45] Yongsen Zang et al. «A column generation tailored to electric vehicle routing problem with nonlinear battery depreciation». En: *Comput. Oper. Res.* 137 (2022), pág. 105527.

5.A. Apéndice

5.A.1. Proposición 5.3

Proposición 5.3. *Sea f una función continua y lineal a trozos, y b una función continua y no decreciente. Defina una función G en un dominio $[x, y]$ como*

$$G(t) = \max_{t' \in [b(x), b(t)]} f(t', t), \tag{5.17}$$

donde $[b(x), b(y)] \subseteq dom(f)$. Entonces, G es continua, lineal a trozos y no decreciente.

Demostración. Supongamos que f tiene $|f|$ piezas $f_1, \ldots, f_{|f|}$ en el dominio $[b(x), b(y)]$, y sea $b(x) = u_0, \ldots, u_{|f|} = b(y)$ sus puntos de quiebre. Dado que b es continua y no decreciente, se sigue que existen $x = t_0 < \ldots < t_{|f|} = y$ tales que $b(t_i) = u_i$ para todo $1 \leq i \leq |f|$. Por inducción en $i = 0, \ldots, |f|$ demostraremos que G es continua, lineal a trozos y no decreciente en el dominio $[t_0, t_i]$. El caso base $i = 0$ es trivial. Para el paso inductivo $i + 1$, sea $z = \max\{f(t') \mid t' \in [u_i, u_{i+1}]\}$. Si $z \leq G(t_i)$, entonces $G(t) = G(t_i) = \max\{f(t') \mid t' \in [u_0, u_i]\}$ para todo $t \in [t_i, t_{i+1}]$; claramente G es continua, lineal a trozos y no decreciente en $[t_0, t_{i+1}]$. De lo contrario, dado que f es continua y f_i es lineal, se sigue que f_i es creciente y existe algún $u^* \in [u_i, u_{i+1}]$ tal que $f_i(u^*) = G(t_i) \geq f(u_i)$. Notar, además, que $u^* = b(t^*)$ para algún $t^* \in [t_i, t_{i+1}]$ porque b es continua. Por lo tanto, $G(t) = G(t_i)$ para todo $t \in [t_i, t^*]$ y, dado que b es no decreciente, $G(t) = f_i(b(t)) = \max\{f(t') \mid t' \in [u_0, b(t)]\}$ para todo $t \in [t^*, t_{i+1}]$. Dado que f_i es lineal, concluimos que G es continua, lineal a trozos y no decreciente en $[t_0, t_{i+1}]$. □

Corolario 5.1. *Sea $|G|$ y $|f|$ el número de piezas de G y f en $[x, y]$ y $[b(x), b(y)]$, respectivamente. Entonces, $|G| \leq |f|$.*

Corolario 5.2. *Si b es lineal a trozos, entonces G puede ser calculada en tiempo polinomial. Si las piezas de f y b están ordenadas, entonces G puede ser calculada en tiempo lineal.*

5.A.2. Ejemplo para la función del nivel de batería

La Figura 5.A.1 ilustra la idea de la función del nivel de batería y sus diferentes etapas. Supongamos que existe un camino de dos vértices $p = (i, j)$ recorrido por un EV para $i, j \in V$. Para simplificar, supongamos que no hay ventanas de tiempo ($a_i = a_j = 0$, $b_i = b_j = T$), el vértice $i = o$ es el depósito, y el tiempo de viaje se ilustra en la Figura 5.A.1b, la capacidad de la batería es $B = 10$ y la función de descarga β_{ij} es como se describe en la Figura 5.A.1a. Finalmente, sea $\hat{g}(t)$ la función de recarga no lineal definida en la Figura 5.A.1c.

El camino de un solo vértice $p_0 = (i) = (o)$ tiene una función de nivel de batería constante $\lambda_{p_0}(t) = B$ para todo $t \in [0, T]$, que representa una carga completa al salir del depósito. Luego, la función de nivel de batería λ_p resulta de extender λ_{p_0} sobre el arco (i, j). En este caso, tenemos dos escenarios posibles: (i) $j \in V \setminus V_s$ o (ii) $j \in V_s$.

En el primer escenario, la función de nivel de batería λ_p después de recorrer el arco (i, j) se muestra en la Figura 5.A.1d. Aquí, el segmento rojo indica los momentos en los que el nivel de batería se alcanzó mediante espera. Por ejemplo, el nivel de batería al llegar a j **exactamente** en el tiempo $t = 10$ es 4. Esto ocurre porque se debe partir del vértice i en el tiempo $t' = 9$, y luego el consumo de batería debe ser 6. Sin embargo, notar que $\lambda_p(10) = 8$, ya que el EV puede llegar a j en el tiempo $t' = 6 < 10$ con nivel de batería 8 y luego esperar 4 unidades de tiempo.

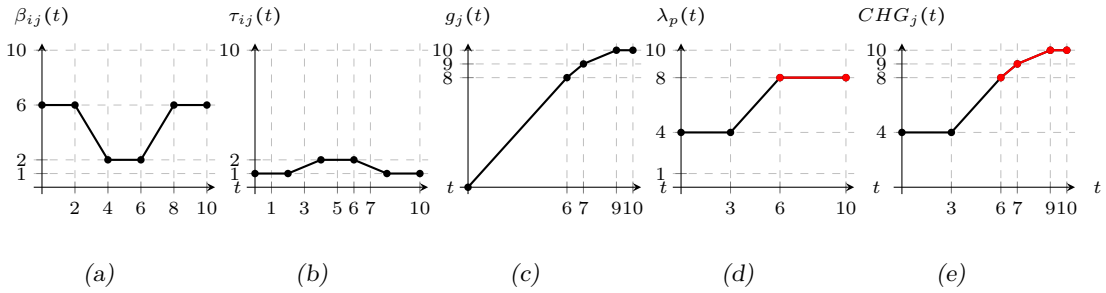


Fig. 5.A.1: Ejemplo de la función de nivel de batería $\lambda_p(t)$, incluyendo la función de descarga β_{ij} (5.A.1a), tiempo de viaje τ_{ij} (5.A.1b), función de carga $g_j(t)$ (5.A.1c), $TRV_{ij}, j \in V \setminus V_s$ (5.A.1d), y $CHG_j, j \in V_s$ (5.A.1e)

El segundo escenario muestra la función de nivel de batería λ_p cuando $j \in V_s$ es una estación de recarga. En este caso, se aplica la función CHG_j y el resultado se ilustra en la Figura 5.A.1e. La porción roja del dominio incluye aquellos tiempos en los que el nivel de batería se alcanzó después de realizar una recarga.

5.A.3. Demostración de la Proposición 5.2

Demostración. Claramente, tener $\bar{y}_r \in \{0,1\}$ para $r \in \Omega$ garantiza que $\bar{z}_{ij} \in \{0,1\}$ por definición, ya que cada arco (i,j) puede estar presente en a lo sumo una ruta con $\bar{y}_r > 0$. Por otro lado, recordar que para cada conjunto de clientes $S \subseteq V_c$ existe a lo sumo una ruta que visita exactamente S en Ω. Adicionalmente, las restricciones (5.12), (5.14) y la hipótesis de que $\bar{z}_{ij} \in \{0,1\}$ imponen que para cada cliente $k \in V_c$ existe exactamente una variable activa $\bar{z}_{kj}$ y $\bar{z}_{ik}$ para algunos $i, j \in V_c \cup \{o,d\}$. Si dos rutas activas $r_1, r_2 \in \Omega$ tales que $\bar{y}_{r_1}, \bar{y}_{r_2} > 0$ comparten un cliente $k \in V_c$, entonces el vértice cliente que sigue a k en $CR(r_1)$ y $CR(r_2)$ debe ser el mismo (o el vértice anterior es el depósito). Un argumento análogo también se aplica para el vértice cliente anterior. Como resultado, es sencillo verificar que $CR(r_1) = CR(r_2)$, y por lo tanto $r_1 = r_2$ ya que existe una ruta única en Ω para cada $S \subseteq V_c$. □

5.A.4. Instancias y Resultados Detallados

La Tabla 5.A.1 muestra los resultados para las dos reglas de branching, CBR y ABR. Reportamos el número de instancias resueltas a optimalidad (opt), el tiempo total de ejecución (time) y el número de nodos enumerados (nodes) agregados por cada política de recarga y cantidad de clientes de la instancia $|V_c|$, y promediados sobre las instancias resueltas por ambos métodos (common).

Los resultados de la comparación con los métodos desarrollados por Desaulniers et al. [7] se muestran en la Tabla 5.A.2, donde la clave es la misma que en los experimentos anteriores. En este caso, los promedios se calculan sobre todas las instancias resueltas de forma óptima por el método correspondiente. Para permitir una comparación, los valores de %rg para los algoritmos desarrollados por Desaulniers et al. [7] se calculan sobre todas las instancias basándose en los resultados que ellos reportan.

			CBR			ABR		
dataset	$\|V_c\|$	common	opt	time	nodes	opt	time	nodes
	25	56	56	2.26	5.29	56	2.59	6.39
SF	50	45	51	76.10	38.11	51	89.48	55.80
	100	10	12	451.64	221.60	12	477.81	273.60
	25	55	56	7.49	5.55	55	9.89	7.76
MF	50	45	49	205.46	25.04	48	176.50	67.84
	100	10	12	509.64	89.40	11	740.99	173.20
	25	56	56	2.08	4.39	56	2.19	6.96
SP	50	48	53	139.21	42.04	50	180.41	66.50
	100	16	17	554.79	45.75	16	661.23	80.75
	25	55	56	8.39	5.22	55	11.62	12.53
MP	50	45	50	247.63	35.80	46	348.45	120.96
	100	9	9	399.76	48.11	9	727.74	92.56

Tab. 5.A.1: Comparación de las diferentes estrategias de branching de la Sección 5.4.3.

			Desaulniers et al. [7]								BCP			
			Monodireccional				Bidireccional							
Var.	$\|V_c\|$	inst	opt	time	nodes	%rg	opt	time	nodes	%rg	opt	time	nodes	%rg
	25	56	55	13.90	1.25	0.08	56	30.60	1.25	0.08	56	2.23	1.36	0.03
SF	50	56	46	265.97	7.74	0.14	47	250.87	7.47	0.16	47	92.28	2.74	0.09
	100	56	13	575.68	33.92	0.12	14	615.30	36.43	0.13	19	552.28	18.16	0.23
	25	56	55	112.98	1.22	0.04	56	16.03	1.21	0.04	56	1.78	1.11	0.01
SP	50	56	44	255.35	4.68	0.09	50	297.38	7.16	0.12	49	162.39	3.12	0.08
	100	56	18	578.65	6.44	0.06	22	628.40	15.36	0.08	22	572.31	12.64	0.12
	25	56	53	93.22	1.19	0.03	56	93.43	1.21	0.03	56	11.08	1.50	0.02
MF	50	56	44	357.85	27.09	0.05	46	258.94	31.17	0.05	46	138.58	1.74	0.04
	100	56	10	599.27	17.20	0.10	13	780.16	77.15	0.14	18	625.86	12.89	0.19
	25	56	52	151.33	1.08	0.01	54	23.48	1.11	0.01	56	7.05	1.04	0.00
MP	50	56	37	374.72	85.43	0.12	45	494.80	82.78	0.12	48	193.27	2.88	0.06
	100	56	15	855.50	5.40	0.10	18	576.32	13.11	0.11	17	775.94	11.82	0.13

Tab. 5.A.2: Comparación con resultados numéricos de Desaulniers et al. [7] para el escenario independiente del tiempo.

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. Conclusiones Generales

Esta tesis aborda diferentes variantes de problemas de ruteo de vehículos con dependencia temporal, que representan de forma más fiel la dinámica de las operaciones de entrega de última milla en áreas urbanas. El trabajo se organiza en torno a tres problemas relacionados: una primera variante modelada mediante una formulación MILP para un solo vehículo; una segunda, resuelta a través de algoritmos especializados de programación dinámica; y una tercera extensión más compleja al caso de múltiples vehículos que incorpora vehículos eléctricos, donde el enfoque se desplaza hacia la estructura del problema y la propuesta de un nuevo modelo matemático.

En el Capítulo 3, se introduce una nueva formulación MILP para el TDTSPTW. El modelo propuesto mantiene las cotas de relajación lineal del estado del arte, al tiempo que reduce significativamente el número de variables y restricciones. Además, se proponen nuevas familias de desigualdades válidas que capturan de forma efectiva la naturaleza temporal del problema. Los resultados computacionales demostraron que este enfoque mejora sustancialmente la capacidad para resolver instancias difíciles, incluyendo algunas que hasta ahora habían permanecido sin resolver.

El Capítulo 4 presenta un algoritmo de programación dinámica basado en *labeling*, específicamente diseñado para entornos con dependencia temporal. Se incorporaron múltiples técnicas de vanguardia, incluyendo estrategias de poda y estructuras de datos eficientes. El algoritmo fue evaluado en cinco conjuntos representativos de benchmarks de la literatura, abarcando variantes como el TDTSP, el TDTSPTW y un TSPTW cuyo objetivo es minimizar la duración total de la ruta. En la mayoría de los casos, el algoritmo superó a los métodos existentes en tiempo de cómputo y número de instancias resueltas.

En el Capítulo 5 se introduce una nueva variante del VRP que combina la dependencia temporal con restricciones operativas propias de los vehículos eléctricos: el TDEVRPTW. Se propone un novedoso algoritmo BCP, que incorpora técnicas modernas para la generación de columnas, desigualdades válidas y una estrategia de *branching* adaptada a este problema. Los resultados experimentales demuestran la competitividad del algoritmo y ponen de manifiesto que ignorar la dependencia temporal puede conducir a decisiones de ruteo ineficientes o incluso inviables.

En conjunto, los hallazgos de esta tesis evidencian cómo las decisiones de modelado —tanto en formulaciones MILP como en algoritmos dinámicos— impactan directamente en la capacidad para resolver instancias prácticas de problemas con tiempos de viaje variables. Las metodologías propuestas amplían el estado del arte en ruteo con dependencia temporal y sientan las bases para abordar escenarios más complejos, como los que involucran vehículos eléctricos y entornos urbanos altamente dinámicos.

Un aprendizaje transversal es que la incorporación de la dependencia temporal incrementa la complejidad algorítmica, pero también magnifica el impacto de las decisiones de implementación. Subestimar estos detalles puede llevar a descartar enfoques que, con las estructuras de datos y estrategias adecuadas, resultarían altamente competitivos.

Asimismo, al igual que en los problemas clásicos con ventanas de tiempo —donde la elección del algoritmo depende de la naturaleza de dichas ventanas (estrechas, amplias, disjuntas)—, en los problemas con tiempos de viaje variables, el desempeño algorítmico está fuertemente influenciado por el perfil de velocidad. Algunas técnicas pueden explotar ciertas regularidades o características del perfil, volviéndose considerablemente más efectivas en contextos específicos.

Finalmente, esta tesis complementa y refuerza líneas de investigación recientes en el área, como las técnicas de *Dynamic Discretization Discovery*, contribuyendo al avance del conocimiento en este tipo particular de problemas. En particular, algunos desarrollos aquí presentados refinan ideas propuestas previamente en la literatura —como en el trabajo de Dabia et al. [1]— al replantear la lógica tradicional de enfoques independientes del problema e incorporar detalles específicos que resultan clave en contextos con dependencia temporal.

6.2. Trabajo Futuro

Las contribuciones de esta tesis abren varias vías prometedoras para la investigación futura en problemas de ruteo de vehículos con tiempos de viaje variables.

En primer lugar, en relación con el TDTSP y el TDTSPTW, sería de interés explorar formulaciones MILP que incorporen restricciones adicionales —como órdenes con prioridad, múltiples ventanas de tiempo o paradas intermedias— sin perder la compacidad que caracteriza al modelo actual.

En cuanto al enfoque de programación dinámica presentado en el Capítulo 4, consideramos que ha sido llevado a un límite. Para lograr un salto cualitativo en eficiencia o capacidad de resolución, será necesario introducir ideas disruptivas que rompan con el paradigma tradicional de *labeling*, permitiendo superar sus barreras estructurales.

Por otro lado, la variante TDEVRPTW presenta múltiples oportunidades de extensión, tanto desde el punto de vista del modelado como del diseño algorítmico. Por ejemplo, incorporar modelos de consumo energético más realistas —como curvas de consumo no lineales que dependan de la velocidad, el peso transportado o la topografía del recorrido— podría mejorar significativamente la fidelidad del modelo. Del mismo modo, considerar estaciones de recarga heterogéneas, con distintas potencias, costos y tecnologías de carga, acercaría el problema a escenarios operativos reales. También resultaría relevante explorar la integración de flotas mixtas (vehículos eléctricos y convencionales), lo que introduce decisiones estratégicas adicionales como la asignación óptima del tipo de vehículo según la ruta. A su vez, podrían incluirse restricciones ambientales como zonas de bajas emisiones o áreas con acceso restringido, que impactan directamente en la planificación de rutas urbanas.

Una línea complementaria consiste en desarrollar enfoques heurísticos que permitan escalar a instancias de mayor tamaño, fundamentales para su aplicación práctica. Estos enfoques podrían actuar tanto sobre la estrategia de resolución como sobre el modelado mismo de la dependencia temporal, proponiendo representaciones más simples o aproximadas que conserven las características clave del fenómeno, y habiliten algoritmos más eficientes. En esta dirección, el uso de metaheurísticas combinadas con componentes exactos representa un camino particularmente prometedor.

Por último, una línea de trabajo fundamental radica en la integración de datos de tráfico del mundo real para validar y enriquecer los modelos desarrollados. La dependencia temporal en los problemas de ruteo está fuertemente vinculada al comportamiento dinámico de las redes de transporte, por lo que utilizar datos empíricos permitiría evaluar con mayor fidelidad la aplicabilidad de los enfoques propuestos. El uso de fuentes como Google Maps, OpenStreetMap o datos históricos de plataformas de navegación posibilitaría la generación de instancias con perfiles de velocidad realistas, incluyendo patrones típicos como congestión en horas pico o eventos excepcionales. Esto no solo permitiría construir benchmarks más representativos y exigentes, sino también fomentar la validación empírica de supuestos de modelado y acercar la investigación académica a casos reales de uso. A su vez, podría abrir oportunidades de colaboración con actores del sector logístico o urbano, favoreciendo la transferencia de conocimiento desde la teoría hacia la práctica.

En resumen, los modelos, algoritmos y resultados presentados en esta tesis amplían el conocimiento sobre el ruteo con dependencia temporal, con especial foco en escenarios urbanos y restricciones energéticas. Estas contribuciones sientan una base sólida para futuras investigaciones que busquen reducir la brecha entre la teoría de optimización y los desafíos logísticos del mundo real.