UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Métodos para el problema de ruteo de vehículos y planificación de tripulaciones simultáneos

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

Mauro Lucci

Directora de tesis:	Dra. Paula L. Zabala
Co-Director de tesis:	Dr. Daniel E. Severín
Consejera de estudios:	Dra. Isabel Méndez-Díaz
Lugar de trabajo:	Departamento de Matemática, Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario.

Buenos Aires, 2023

Métodos para el problema de ruteo de vehículos y planificación de tripulaciones simultáneos

Resumen— En las últimas décadas, los problemas de ruteo de vehículos y de asignación de tripulaciones han sido ampliamente estudiados por separado. Recientemente, ha habido un creciente interés por combinarlos en un único problema simultáneo, abandonando la simplificación de que una misma tripulación debía ejecutar la totalidad de una ruta sin posibilidad de ser relevada. El resultado es un problema difícil desde la optimización combinatoria, que involucra complejas restricciones operativas, laborales y de sincronización. En esta tesis se estudian métodos para la resolución heurística y exacta de este problema, siguiendo el caso de estudio de una empresa real que debe cumplir con pedidos de recolección y entrega de mercadería con ventanas de tiempo en larga distancia, minimizando distancias recorridas y demoras en las entregas. Por un lado, se analiza una descomposición secuencial en dos etapas, realizando primero el ruteo de los vehículos y posteriormente la asignación de las tripulaciones sobre segmentos de rutas. Para cada etapa, se desarrollan algoritmos basados en metaheurísticas híbridas. Por otro lado, se proponen modelos de Programación Lineal Entera que resuelven el problema en una única etapa y se estudian familias de desigualdades válidas. Todos estos desarrollos se incorporan luego en un algoritmo de branch-and-cut.

Palabras claves— Ruteo de vehículos, Planificación de tripulaciones, Metaheurísticas, Programación Lineal Entera, Branch-and-cut.

Methods for the simultaneous vehicle routing and crew scheduling problem

Abstract— In recent decades, vehicle routing and crew scheduling problems have been widely studied separately. Recently, there has been growing interest in combining them into a single simultaneous problem, abandoning the simplification that the same crew must execute an entire route without the possibility of being relieved. The result is a hard combinatorial optimization problem, involving complex operational, labor, and synchronization constraints. In this thesis, methods are studied for the heuristic and exact resolution of this problem, following the case study of a real company that must fulfill requests for the pickup and delivery of goods in long-distance road transport, minimizing travel distances and delays in deliveries. On the one hand, a two-stage sequential decomposition is analyzed, first performing vehicle routing and subsequently assigning crews to route sections. For each stage, algorithms based on hybrid metaheuristics are developed. On the other hand, Integer Linear Programming models are proposed to solve the problem in a single stage and families of valid inequalities are studied. All these developments are later incorporated into a branch-and-cut algorithm.

Keywords— Vehicle routing, Crew scheduling, Metaheuristics, Integer Linear Programming, Branch-and-cut.

Agradecimientos

Está lejos de ser una tarea sencilla poder agradecer en tan pocas líneas a tantas personas que me acompañaron en este recorrido, no tengo dudas de que todo fue posible gracias a ellas.

A mis directores Paula y Daniel, por darme la oportunidad de trabajar con ellos y por compartir conmigo su tiempo y sabiduría.

A mamá, papá, Anto, Fede, nona Dina y nona Rogel, por su amor sin medidas, por brindarme todo lo que estuvo siempre a su alcance y apoyarme en cada una de mis decisiones, porque de ser posible los elegiría una y mil veces.

A Nahue, por acompañarme desde el inicio del doctorado, por los infinitos momentos que compartimos, por ser mi soporte cuando los días pesaban y porque seguimos eligiendo crecer juntos.

A Mela y Tito, por invitarme a almorzar en mis primeros días como becario, incluirme en el grupo y hacer posible que el lugar de trabajo sea también mi casa, pero por sobre todas las cosas, porque me enseñaron lo linda que es la vida cuando sos auténtico. A Grace, por tantos momentos compartidos en la oficina, los mates, las eternas charlas de juegos de mesa, las juntadas a pesar del cansancio. A todos los amigos que conocí en la facu, Ana, Dai, Delfi, Esponja, Giuli, Ger, Grace, Gret, Mela, Nacho, Nahue, Tito, Smart y Sofi, por las risas diarias en el comedero, las incontables noches de juegos de mesa, baratijas y variedades de joyas bebibles, porque rodearse de personas así hace que el mundo sea un lugar mejor.

A mis amigos de Casilda, Aldi S., Aldi P., Alvi, Bona, Caco, Ceci, Dipra, Eri, Fabri, Facu, Feli, Ferru, Giudi, Juanchi, Leo, Luchi, Lucho, Mati, Neri, Nico, Niño, Sampa, Sofi, Tadich, Tincho, Tiño, Tío y Tkb, por las miles de juntadas, los asados y las salidas, porque me vienen acompañando desde toda mi vida.

A Gra por iniciarme en la Optimización Combinatoria, enseñarme lo apasionante del área, compartir su enorme experiencia y motivarme a seguir creciendo.

A Fede por estar siempre presente, escuchar e involucrarse, por tantas invitaciones de café que no me alcanza la vida para devolver.

Dedicado al gran Tata,
y al querido Benja,
eternamente conmigo.

Índice general

1. Introducción

En este capítulo introductorio se presenta la motivación que lleva al estudio de métodos para el problema de ruteo de vehículos y planificación de tripulaciones simultáneos y se realiza un repaso del estado del arte sobre importantes problemas relacionados del área.

El desarrollo de la tesis sigue un caso de estudio real proveniente de una empresa de transporte que se ocupa de la distribución de café en Colombia. Tanto el caso de estudio como la notación usada para referir a sus elementos se introducen también en este capítulo. Los requerimientos operativos a los cuales se hace mención, como así también las regulaciones laborales, son fieles a lo planteado por la empresa.

Antes de continuar, se sugiere a los lectores no experimentados con temas del área revisar los apéndices del Capítulo 9. Allí se definen conceptos preliminares necesarios para la comprensión de la tesis, que incluyen nociones de Teoría de Grafos, Optimización Combinatoria y Complejidad Computacional, Programación Lineal Entera, Heurísticas y Metaheurísticas y Optimización Multiobjetivo.

1.1. Problemas de Ruteo de Vehículos

Se conoce como *Problemas de Ruteo de Vehículos (VRPs[1], véase [Toth and Vigo, 2014])* a la familia de problemas con la siguiente descripción coloquial:

[1]Del inglés, Vehicle Routing Problems

Entrada. Un conjunto de *pedidos de transporte* y una *flota de vehículos*.

Objetivo. Encontrar un conjunto de *rutas de vehículo* para la flota que *cumpla* con todos los (o algunos) pedidos a *mínimo costo*; lo que involucra asignar una secuencia de pedidos a cada vehículo de manera que cada ruta se pueda ejecutar de forma *factible*.

En esta definición se resaltan las principales componentes de los VRPs: el tipo de pedidos de transporte y la forma de realizarlos, la flota de vehículos, los costos a minimizar y la factibilidad de las rutas. Las características de cada una de estas componentes dan lugar a diferentes versiones y variantes del problema.

En 1959, Dantzig y Ramser introdujeron el primer VRP mediante una aplicación de la vida real referida al abastecimiento de combustible a estaciones de servicio [Dantzig and Ramser, 1959]. Desde este trabajo fundacional hace más de 60 años, una enorme cantidad de artículos han sido publicados, presentado modelos matemáticos y algoritmos exactos y (meta)heurísticos para la resolución óptima y aproximada de VRPs. El gran interés de la comunidad científica internacional y de los mercados por estos problemas se puede explicar en su alto impacto económico y en su cautivadora complejidad desde la perspectiva de la optimización combinatoria.

En la actualidad, muchos de los algoritmos de optimización desarrollados para la resolución de VRPs han sido incorporados en herramientas de software comerciales, quienes desempeñan un papel protagónico en la búsqueda de soluciones factibles de alta calidad. Mediante estas herramientas es posible resolver grandes instancias del mundo real en tiempos de cómputo razonables, lo que significa ahorros substanciales en los costos del transporte. El éxito no es mera consecuencia del aumento del poder de cómputo de los sistemas informáticos modernos, sino también, de los profundos estudios científicos que derivaron en modelos matemáticos lo suficientemente ricos para capturar gran variedad de características surgidas en aplicaciones del mundo real. Sin embargo, las diferencias existentes en las legislaciones laborales de cada país, en las políticas propias de cada empresa o en los acuerdos colectivos de trabajo de cada sindicato, pueden muchas veces limitar o imposibilitar la aplicabilidad de estos software.

En las versiones más estudiadas de VRPs, los pedidos de transporte consisten en distribuir mercancías desde un depósito y entregarlas a un conjunto de clientes (*Problemas de Última Milla*), e.g. el abastecimiento de combustible a estaciones de servicio, o en recolectar mercancías de un conjunto de clientes y llevarlas a un depósito (*Problemas de Primera Milla*), e.g. la recolección de leche cruda de los tambos. En cambio, muchas aplicaciones requieren transportes *punto-a-punto*, esto es, llevar mercancías desde un punto de recolección hacia otro de entrega, y generalmente ninguno de ellos es un depósito. Esta variante se conoce como *Problema de Recolección y Entrega (PDP[2])* y recibe un nombre diferente cuando se trasladan personas (*Dial-a-Ride Problem*).

Un aspecto fundamental cuando se definen variantes de VRPs es el tipo de restricciones que condicionan la factibilidad de una ruta de vehículo. Un ejemplo típico es cuando las decisiones de ruteo están acopladas con decisiones de *scheduling*. En el *Problema de Ruteo con Ventanas de Tiempo (VRPTW[3])* o en el *Problema de Recolección y Entrega con Ventanas de Tiempo (PDPTW[4])*, cada pedido de transporte tiene asociado una *ventana de tiempo* para su cumplimiento. En este contexto, determinar si una ruta de vehículo es factible requiere verificar si es posible asignar una hora de inicio a cada pedido de forma tal que (i) cada pedido se cumpla dentro de su ventana de tiempo y (ii) si dos pedidos son consecutivos en una ruta, entonces el segundo comienza luego de que el primero termine más el tiempo de viaje entre ellos.

Particularmente en el transporte de larga distancia, donde los vehículos suelen estar fuera del depósito durante varios días, se vuelve imprescindible que las rutas de los vehículos contemplen el descanso de los conductores[5]. Las restricciones sobre las horas de trabajo y los periodos de descanso son probablemente el tipo de restricciones de scheduling más complejo. Las variantes de VRPTW y PDPTW en cumplimiento con regulaciones laborales suelen ser extremadamente desafiantes, donde incluso el simple hecho de verificar la factibilidad de una ruta puede ser una tarea altamente compleja en sí misma.

[2]Del inglés, Pickup and Delivery Problem
[3]Del inglés, Vehicle Routing Problem with Time Windows
[4]Del inglés, Pickup and Delivery Problem with Time Windows
[5]A pesar de referirlos en masculino, también se incluyen a conductoras y a otras identidades de género.

En la gran mayoría de literatura referida a VRPs, los vehículos y los conductores se asumen inseparables, es decir, un conductor está ligado a un vehículo y debe ejecutar la totalidad de su ruta. En consecuencia, las regulaciones laborales de los conductores se imponen innecesariamente sobre los vehículos. En el *Problema de Ruteo de Vehículos y Planificación de Tripulaciones Simultáneos (SVRCSP[6])*, se permiten intercambios de conductor-camión a lo largo del horizonte de planificación. Abandonando la correspondencia biunívoca entre vehículos y conductores, es posible aumentar la productividad, dado que un nuevo conductor puede continuar con el transporte cuando el anterior descanse (de hecho, toda solución donde los vehículos y los conductores no se intercambian es también solución para la nueva variante). Como contrapartida, el problema se vuelve aún más complejo dado que ahora es necesario encontrar rutas separadas para los vehículos y para los conductores, sujetas a fuertes restricciones de sincronización espacio-temporales. Además, la sincronización genera interdependencia entre las rutas e incluso una pequeña modificación en una ruta puede afectar la factibilidad de la demás; por ejemplo, demorar un viaje puede generar que el conductor no sea capaz de subirse a otro vehículo que lo estaba esperando (este problema se introduce en [Drexl, 2012]).

1.2. Caso de estudio

Este trabajo se enfoca en un escenario donde se debe realizar una planificación de múltiples días (típicamente una semana) de los camiones y los conductores de una compañía, para cumplir con un conjunto de pedidos de recolección y entrega de mercadería en larga distancia y con ventanas de tiempo, optimizando los costos operativos y la demora en las entregas.

Cada pedido involucra la recolección (o carga) de una mercadería en alguna localidad inicial, su transporte hacia otra localidad de destino y su posterior entrega (o descarga). Todos los pedidos deben estar entregados antes de que finalice el horizonte de planificación. El mismo camión que recolecta un pedido debe ser el encargado de entregarlo, es decir, no se permiten *transbordos* de la mercadería.

[6]Del inglés, Simultaneous Vehicle Routing and Crew Scheduling Problem. La nomenclatura no está todavía del todo universalizada.

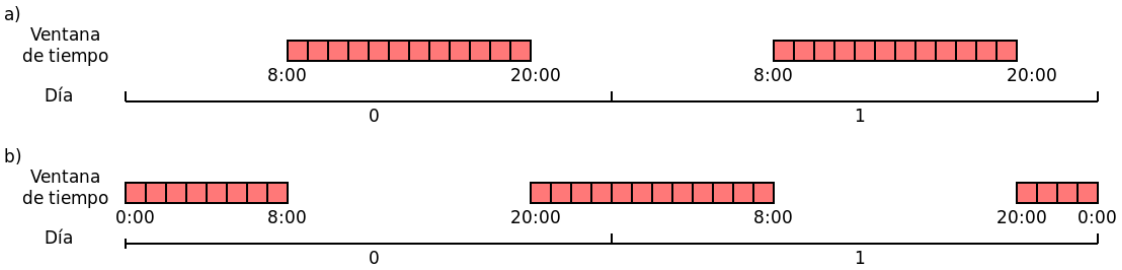


Figura 1.2.1.: Ejemplos de ventanas de tiempo.

Además, los camiones pueden transportar un único pedido a la vez. El tiempo que demora la carga/descarga de una mercadería en el camión, conocido como *tiempo de servicio*, es homogéneo, siendo en nuestro caso de 1 hora, mientras tanto el conductor debe permanecer en el vehículo. La recolección de cada pedido tiene asociados un día inicial y una ventana de tiempo, los cuales restringen el día y el horario en el que puede empezar la carga de la mercadería, aunque la misma podría finalizar fuera de dicha ventana. De manera análoga, la entrega de cada pedido también tiene asociados su propio día inicial y ventana de tiempo. Estas restricciones representan el día a partir del cual el cliente espera el servicio y el horario de atención que maneja. Las ventanas de tiempo se mantienen abiertas desde una hora inicial hasta una hora final (que puede ser al día siguiente siempre que no exceda las 24 horas de duración) y repiten diariamente este comportamiento desde su día inicial hasta el último día del horizonte de planificación. A su vez, las ventanas de tiempo son *rígidas*, es decir, un camión que llega antes de tiempo, o incluso antes del día inicial, debe esperar a que la ventana abra para poder comenzar con el servicio. En la Figura 1.2.1.a se ejemplifica una ventana de tiempo que abre a partir del primer día desde las 8:00 hasta las 20:00; mientras que en 1.2.1.b, una que abre desde las 20:00 hasta las 8:00 del día siguiente, donde cada cuadrado representa 1 hora del horizonte. Notar que en el segundo caso, la ventana abre parcialmente de 0:00 a 8:00 del día inicial y de 20:00 a 0:00 del último día. Los pedidos no tienen plazos de entrega, es decir, ellos pueden entregarse en cualquier día a partir de su día inicial. Sin embargo, existe una penalidad por demora asociada a cada pedido, la cual es proporcional al número de días transcurridos desde el día en que la ventana de tiempo de entrega abre por primera vez y el día en que efectivamente se concreta la entrega.

Los camiones y conductores de la compañía son *homogéneos*. Esto quiere decir que,

a los fines del problema, ellos son iguales entre sí. Entre otras cosas, cada camión tiene el mismo consumo de combustible, la capacidad suficiente para transportar cualquiera de los pedidos en un único viaje, cada conductor puede conducir cualquiera de los camiones y obedece la misma regulación laboral. A diferencia de otros problemas donde los camiones y conductores se ubican inicialmente en un depósito central, aquí cada uno de ellos tiene su propia localidad inicial desde donde arranca al comienzo del horizonte de planificación. No obstante, ellos no deben regresar a su localidad inicial al finalizar el horizonte, sino que la localidad final pasará a ser inicial en la próxima planificación. Existe un conjunto de localidades donde los camiones y conductores están autorizados a detenerse, y no pueden hacerlo en otros lugares, por ejemplo al costado de una ruta. Este conjunto incluye además las localidades de recolección y de entrega de los pedidos y las localidades iniciales de los camiones y los conductores. La distancia de viaje entre las mismas es conocida y el tiempo de viaje es constante a lo largo del día.

Los conductores son los únicos que pueden manejar los camiones de la compañía, pero además ellos pueden trasladarse solicitando un servicio de taxis externo por un costo adicional. Así, la planificación de las tripulaciones se vuelve más flexible, dado que los conductores pueden usar este servicio para subirse a un camión que se encuentra en otra localidad. Hay un número ilimitado de taxis disponibles en cualquier localidad y en cualquier momento, pero cada uno de ellos puede trasladar a lo sumo a una persona a la vez. Siempre que un conductor esté viajando (en camión o en taxi) y durante los tiempos de servicio, se considera que está *trabajando*; y en cualquier otro momento, *descansando*. Un descanso comienza cuando el conductor desciende de un vehículo y termina cuando sube al mismo o a otro vehículo (camión o taxi). En general, los descansos están regulados por leyes laborales, acuerdos colectivos de trabajo y políticas internas de la compañía. En este caso, los conductores deben descansar al menos 12 horas en cada intervalo de 24 horas y al menos un día completo en cada intervalo de 7 días. Así expresadas, resultan más restrictivas que pedir un descanso por día y un franco por semana, dado que evitan por ejemplo que un conductor trabaje sin descanso las últimas 12 horas del primer día y las primeras 12 horas del día siguiente, como así también, que trabaje en los últimos 6 días de la primera semana y en los primeros 6 días de la semana siguiente. Por otro lado, los conductores pueden cambiar libremente de camión en cualquier localidad autorizada y en cualquier momento (antes, durante o después

de transportar una mercadería). De esta forma, es posible evitar los tiempos muertos que los camiones heredan de los descansos de los conductores. Por ejemplo, un conductor que necesite descansar puede desviar el camión a la localidad más cercana, relevar al conductor por otro ya descansado e inmediatamente continuar con el transporte de la mercadería sin interrupciones. Un conductor puede bajar de un camión a descansar o puede inmediatamente subir a un vehículo diferente (camión o taxi).

Una de las particularidades de este problema es que las tripulaciones de los camiones pueden estar compuestas por uno o por dos conductores. En este último caso, ambos se consideran trabajando (a pesar de que sólo uno de ellos esté manejando) y pueden ser relevados de manera independiente uno del otro. Esta posibilidad incorpora aún más combinatoria a la planificación, pero busca evitar el uso de taxis en determinadas circunstancias. Por ejemplo, un camión puede desviar su recorrido para ir a buscar a un segundo conductor y dejarlo en otra localidad antes de continuar con su ruta. Por supuesto, decidir si esto resulta redituable frente a la opción de solicitar un taxi dependerá de los costos asociados. En este sentido, el problema involucra optimizar los siguientes objetivos: minimizar el costo por los kilómetros recorridos en camión y en taxi y minimizar la penalidad por las demoras en las entregas de las mercaderías.

El caso de estudio propuesto se basa en un escenario real que involucra a una empresa dedicada al transporte de café en Colombia. Desafortunadamente, no fue posible establecer una colaboración directa en esta investigación, por lo que no se mencionará el nombre de la empresa en este trabajo. Sin embargo, la descripción proporcionada se ajusta fielmente a los requisitos originales planteados por la empresa.

1.2.1. Notación

En el siguiente listado, se encuentra la notación que será usada a lo largo de la tesis para referir a los distintos componentes mencionados en el caso de estudio.

- H: número de días del horizonte de planificación.
- I: número de instantes de tiempo por día, bajo alguna discretización.

- $\mathcal{I} = \{0, \ldots, I.H\}$: conjunto de instantes de tiempo en el horizonte, $i \in \mathcal{I}$ instante de tiempo.
- C: conjunto de conductores, $c \in C$: conductor.
- V: conjunto de camiones, $v \in V$: camión.
- L: conjunto de localidades, $l \in L$: localidad.
- l_v y $l_c \in L$: localidades iniciales del camión $v \in V$ y del conductor $c \in C$, respectivamente.
- $L_V = \{l_v : v \in V\}$ y $L_C = \{l_c : c \in C\}$.
- P: conjunto de pedidos, $p \in P$: pedido con
  - l_p^{R} y $l_p^{\mathrm{E}} \in L$: localidades de recolección y de entrega de p, respectivamente.
  - dia_p^{R} y $dia_p^{\mathrm{E}} \in \{0, \ldots, H-1\}$: días iniciales para la recolección y la entrega de p, respectivamente.
  - a_p^{R} y $b_p^{\mathrm{R}} \in \{0, \ldots, I-1\}$ con $a_p^{\mathrm{R}} \neq b_p^{\mathrm{R}}$: instantes de tiempo (diarios) en que abre y cierra la ventana de tiempo de recolección de p, respectivamente.

    Si $a_p^{\mathrm{R}} < b_p^{\mathrm{R}}$, entonces la ventana abre y cierra en el mismo día. Por el contrario, si $a_p^{\mathrm{R}} > b_p^{\mathrm{R}}$, entonces la ventana cierra al día siguiente.

    Análogamente, se nota con a_p^{E} y $b_p^{\mathrm{E}} \in \{0, \ldots, I-1\}$ a los instantes de tiempo en que abre y cierra la ventana de tiempo de entrega de p, respectivamente.
  - s_p^{R} y s_p^{E}: tiempos de servicio para la recolección y la entrega de p medidos en instantes de tiempo, respectivamente. En las instancias consideradas en esta tesis, siempre serán iguales a 1.
  - w_p: penalidad por día de demora en la entrega.

- $\mathcal{I}_p^{\mathrm{R}}$ y $\mathcal{I}_p^{\mathrm{E}} \subset \mathcal{I}$: subconjuntos de instantes de tiempo donde puede comenzar la recolección y la entrega de un pedido $p \in P$, respectivamente.

  Por ejemplo, en la Figura 1.2.1, si la ventana de tiempo del caso a) corresponde a la recolección de un pedido p, entonces $\mathcal{I}_p^{\mathrm{R}} = \{8, \ldots, 20\} \cup \{32, \ldots, 44\}$. Si el caso b) corresponde a la entrega de un pedido p, entonces $\mathcal{I}_p^{\mathrm{E}} = \{0, \ldots, 8\} \cup \{20, \ldots, 32\} \cup \{44, \ldots, 48\}$.

- dia : $\mathcal{I} \to \{0, \ldots, H-1\}$: función que dado un instante de tiempo i retorna el día del horizonte al que pertenece, es decir, dia(i) es la división entera de i por I.

- hora : $\mathcal{I} \to \{0, \ldots, I-1\}$: función que dado un instante de tiempo i retorna el instante de tiempo (diario) al que se corresponde, es decir, hora(i) es el resto de dividir a i por I.

- tiempoViajeCamion y tiempoViajeTaxi: funciones que dadas dos localidades retornan el número de instantes de tiempo requeridos para viajar entre dichas localidades en camión y en taxi. De ser necesario, estas funciones se abreviarán como tVC y tVT, respectivamente.

- costoViajeCamion y costoViajeTaxi: funciones que dadas dos localidades retornan el costo de viajar entre dichas localidades en camión y en taxi, respectivamente.

1.2.2. Ejemplo

Se introduce a continuación un ejemplo que será referenciado a lo largo de la tesis para ilustrar diferentes conceptos desarrollados en la misma.

Ejemplo 1.2.1. Se considera una instancia del problema que involucra dos localidades l_1 y l_2 y un horizonte de planificación de 1 día discretizado cada 3 horas, es decir, $L = \{l_1, l_2\}$, $H = 1$, $I = 8$ e $\mathcal{I} = \{0, \ldots, 8\}$. El tiempo de viaje en camión y en taxi entre ambas localidades es de 3 horas (1 instante de tiempo), es decir, $\mathrm{tVC}(l_1, l_2) = \mathrm{tVC}(l_2, l_1) = \mathrm{tVT}(l_1, l_2) = \mathrm{tVT}(l_2, l_1) = 1$. Se dispone de un camión v_1 y un conductor c_1 en la localidad inicial l_1 y de un camión v_2 y un conductor c_2 en l_2, es decir, $l_{v_1} = l_{c_1} = l_1$ y $l_{v_2} = l_{c_2} = l_2$.

Se tiene un conjunto de pedidos $P = \{p_1, p_2\}$, definidos con los siguientes atributos.

Pedido p_1:

$$dia^{\mathrm{R}}_{p_1} = dia^{\mathrm{E}}_{p_1} = 0$$
$$[a^{\mathrm{R}}_{p_1}, b^{\mathrm{R}}_{p_1}] = [0, 2] \text{ (0:00-6:00)}$$
$$[a^{\mathrm{E}}_{p_1}, b^{\mathrm{E}}_{p_1}] = [6, 2] \text{ (18:00-6:00)}$$
$$l^{\mathrm{R}}_{p_1} = l_1$$
$$l^{\mathrm{E}}_{p_1} = l_2$$
$$s^{\mathrm{R}}_{p_1} = s^{\mathrm{E}}_{p_1} = 1 \text{ (3 h)}$$

Pedido p_2:

$$dia^{\mathrm{R}}_{p_2} = dia^{\mathrm{E}}_{p_2} = 0$$
$$[a^{\mathrm{R}}_{p_2}, b^{\mathrm{R}}_{p_2}] = [3, 5] \text{ (9:00-15:00)}$$
$$[a^{\mathrm{E}}_{p_2}, b^{\mathrm{E}}_{p_2}] = [3, 5] \text{ (9:00-15:00)}$$
$$l^{\mathrm{R}}_{p_2} = l_2$$
$$l^{\mathrm{E}}_{p_2} = l_1$$
$$s^{\mathrm{R}}_{p_2} = s^{\mathrm{E}}_{p_2} = 1 \text{ (3 h)}$$

Además, se va a considerar el siguiente plan de rutas.

Camión v_1 (inicia en l_1):

0:00–3:00	Carga p_1.
3:00–6:00	Viaja de l_1 a l_2.
6:00–9:00	Descarga p_1.
9:00–0:00	Espera en l_2.

Camión v_2 (inicia en l_2):

0:00-9:00	Espera en l_2.
9:00-12:00	Carga p_2.
12:00-15:00	Viaja de l_2 a l_1.
15:00-18:00	Descarga p_2.
18:00-0:00	Espera en l_1.

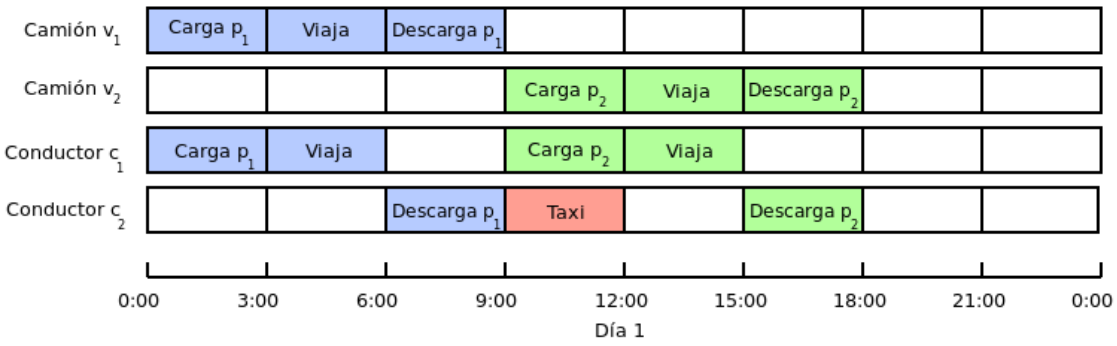
Conductor c_1 (inicia en l_1):

0:00-3:00	Carga p_1.
3:00-6:00	Viaja de l_1 a l_2 en v_1.
6:00-9:00	Descansa en l_2.
9:00-12:00	Carga p_2.
12:00-15:00	Viaja de l_2 a l_1 en v_2.
15:00-0:00	Descansa en l_2.

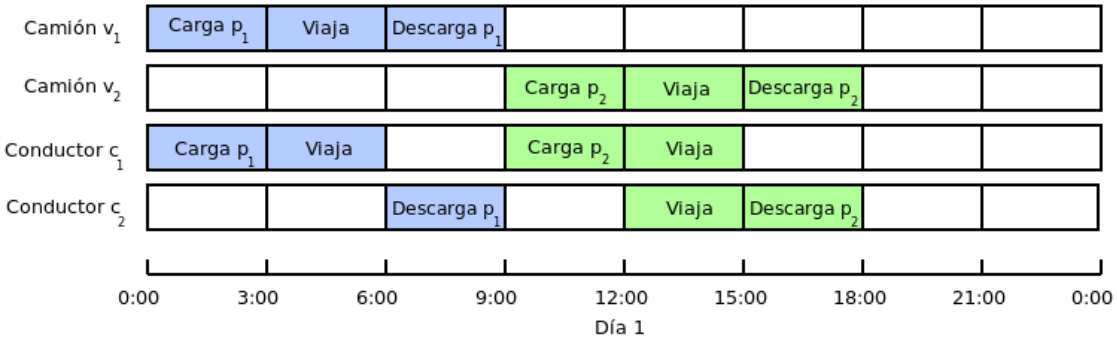
Conductor c_2 (inicia en l_2):

0:00-6:00	Descansa en l_2.
6:00-9:00	Descarga p_1.
9:00-12:00	Viaja de l_2 a l_1 en taxi.
12:00-15:00	Descansa en l_1.
15:00-18:00	Descarga p_2.
18:00-0:00	Descansa en l_1.

Un esquema de este plan de rutas puede observarse en la Figura 1.2.2a, donde el color de la tarea identifica el camión que la realiza. Particularmente la entrega de p_1 se puede realizar en ese instante de tiempo dado que su ventana de tiempo de entrega abre y cierra al día siguiente, por lo tanto, $\mathcal{I}^{\mathrm{E}}_{p_1} = \{0, 1, 2\} \cup \{6, \ldots, 8\}$, tal como había sido explicado en la Figura 1.2.1.b.



(a)



(b)

Figura 1.2.2.: Ejemplos de planes de ruta.

Vale la pena mencionar que el plan de rutas propuesto no es óptimo, aunque para los fines de este ejemplo no será relevante. Una mejor solución se obtendría notando que si el conductor c_2 toma su descanso antes de viajar a localidad l_1, entonces podría ahorrarse el costo del taxi compartiendo el viaje en el camión v_2 con el conductor c_1. Un esquema de dicho plan de rutas puede observarse en la Figura 1.2.2b.

1.3. Antecedentes

El problema abordado en esta tesis fue estudiado previamente en [Bakarcic et al., 2013], en el marco de una tesis de grado bajo la dirección de las Dras. Méndez-Díaz y Zabala, aunque cabe aclarar que en dicha investigación no se incluyó un servicio de taxis externo a la compañía. Los autores propusieron un modelo de *Programación Lineal Entera* (PLE) con una cantidad exponencial de variables (hay una variable por cada posible ruta). Lo resuelven por medio de generación de columnas, es decir, comienzan con un conjunto restringido de columnas e iterativamente agregan columnas con costo reducido negativo hasta que la relajación lineal sea resuelta por optimalidad o hasta alcanzar un determinado criterio de parada. Luego, el programa resultante (con las columnas que hayan sido generadas) se resuelve con un solver de PLE de propósito general. Este enfoque es en efecto heurístico dado que la generación de columnas se aplica únicamente en el nodo raíz del árbol de decisión. De esta forma, logran obtener buenos resultados en instancias con 6-7 localidades y 6-12 pedidos en 900 segundos de tiempo de ejecución, pero las soluciones obtenidas para instancias con 10 localidades y 16 pedidos tras 8 horas de ejecución no mejoran las obtenidas con una heurística golosa. Hasta donde llega nuestro conocimiento, este problema no fue estudiado por otros autores.

En lo que resta de la sección, se hace un repaso histórico sobre las principales investigaciones en otros problemas de ruteo de vehículos relacionados, en mayor o menos medida, con el caso de estudio.

Una amplia variedad de regulaciones sobre conductores han sido consideradas en problemas de planificación de tripulaciones para otros sectores del transporte, e.g. para líneas aéreas desde los años 1960 [Arabeyre et al., 1969]. Un trabajo pionero en la integración de ruteo y planificación en el sector aéreo es [Cordeau et al.,

2001], que aborda un problema simultáneo de ruteo de aviones y planificación de tripulaciones. Allí, se contemplan diferentes restricciones, tales como tiempo máximo de vuelo, requerimientos de mantenimiento, tiempos de conexión mínimos para tripulaciones que necesiten conexiones aéreas y periodos de descanso. En [Salazar-González, 2014], se aborda de forma simultánea un problema que involucra ruteo de aeronaves, emparejamiento de tripulaciones y planificación de flotas para una aerolínea regional que opera entre las Islas Canarias. Es posible encontrar problemas similares en el ámbito del transporte público (ver e.g. [Haase et al., 2001, Ciancio et al., 2018]).

La planificación para estos sectores del transporte es muy diferente, dado que en general operan con viajes ya programados, los cuales nunca se interrumpen para que el personal descanse. En el transporte de cargas de larga distancia, los clientes pueden ser atendidos en cualquier momento dentro de sus ventanas de tiempo y los viajes pueden, y en general deben, interrumpirse cuando el conductor deba tomar su descanso reglamentario. En consecuencia, el tiempo de viaje entre localidades no solo depende de la distancia, sino también de las horas trabajadas previamente por el conductor y de los descansos intermedios que necesite.

Hasta los años 2000, las regulaciones sobre los conductores recibieron muy poca atención en el transporte de cargas de larga distancia. Los primeros trabajos en incluir este tipo de regulaciones son probablemente [Savelsbergh and Sol, 1998] y [Xu et al., 2003]. El primero de ellos, contempla pausas para almorzar y descansos nocturnos para los conductores, mientras que el segundo, sigue la legislación laboral para conductores prescrita por el Departamento de Transporte de EEUU. Entre los enfoques heurísticos para las regulaciones de la UE (Unión Europea), se destacan los propuestos en [Goel, 2009, Kok et al., 2010, Prescott-Gagnon et al., 2010]. En particular, el último de ellos propone una *Large Neighborhood Search* (LNS), donde los vecindarios se exploran usando una heurística de generación de columnas que aplica una Búsqueda Tabú para generar nuevas columnas. Entre los trabajos que contemplan las regulaciones de EEUU, se destaca [Rancourt et al., 2013], quienes desarrollaron una Búsqueda Tabú que embebe varios algoritmos de planificación. Por su parte, [Goel and Vidal, 2014] desarrollaron un enfoque globalizador capaz de manejar diferentes regulaciones, entre ellas, las de EEUU, la UE, Canadá, y Australia, mediante una Búsqueda Genética híbrida. Recientemente, [Koç et al., 2018] proponen una metaheurística de arranque múltiple, que combina

una LNS adaptativa y PLE, para resolver una extensión del problema que incorpora costos asociados al suministro de energía eléctrica, requerida para mantener el confort de los conductores cuando el camión está estacionado. Con respecto a métodos exactos, [Goel and Irnich, 2017] proponen un algoritmo *Branch-and-Cut* (B&C), convirtiéndose así en los primeros en resolver este problema por optimalidad. Posteriormente, [Tilk and Goel, 2020] desarrollaron un algoritmo *Branch-and-Price-and-Cut*, el cual fue evaluado con la regulación de la UE y la de EEUU. Recientemente, algunos autores han comenzado a indagar la posibilidad de que dos conductores puedan viajar juntos en un mismo camión. Por ejemplo, [Goel et al., 2021] sugieren que ni la conducción individual ni la conducción en equipos son buenas prácticas por sí solas, sino que una mezcla de ambas puede resultar en una mejora sobre los costos operativos. En general, todos estos métodos han sido evaluados sobre el conjunto de instancias *benchmark* propuesto por [Goel, 2009], que cuenta con 100 clientes (o pedidos) y un horizonte de planificación de 144 horas (6 días).

Un subproblema clave con el que deben lidiar los enfoques anteriores es el de decidir si un conductor puede ejecutar la ruta de un camión particular. Es decir, dada la secuencia de pedidos que realiza un camión con sus ventanas de tiempo, decidir si existe alguna asignación de horarios para los pedidos de modo que un conductor pueda realizarlos secuencialmente cumpliendo con las ventanas de tiempo y con las regulaciones laborales. Incluso la resolución de este subproblema, el cual es más acotado porque interviene un único camión y el orden de los pedidos ya está determinado, puede llegar a ser extremadamente desafiante, como lo ilustran los siguientes trabajos. Para la regulación de EEUU anterior a la modificación de 2013, el problema era resoluble en tiempo polinomial para ventanas de tiempo simples y múltiples suficientemente distanciadas [Archetti and Savelsbergh, 2009, Goel and Kok, 2012]. Recientemente, [Kleff, 2019] mostró que es posible resolverlo en tiempo polinomial cuando las regulaciones incluyen dos tipos de descansos (esto incluye la de EEUU y la UE) y los descansos se toman únicamente en las localidades de los clientes. Sin esta asunción, la existencia de un algoritmo de tiempo polinomial es aún una pregunta abierta, al igual que para las regulaciones de otros países.

Todos estos trabajos recién citados asumen que los mismos conductores operan la totalidad de la ruta de un camión. Además, los equipos de conductores se asumen unidades indivisibles, cuyo beneficio viene de regulaciones laborales especiales, por

ejemplo, un camión puede moverse de forma casi ininterrumpida por un total de 18 horas al ser conducido por un equipo de dos conductores. Por el contrario, en el problema considerado en este trabajo, los conductores pueden cambiar de camión durante su jornada laboral y los equipos pueden dividirse. A pesar de que los SVRCSP han sido muy poco estudiados, es posible encontrar en la literatura algunos trabajos referenciales. A continuación, se mencionan los más cercanos a esta investigación.

[Drexl et al., 2013] resuelve un PDPTW en larga distancia, sujeto a la regulación de conductores de la UE. Los cambios de camión-conductor pueden ocurrir en múltiples depósitos, llamados estaciones de relevo, pero el enfoque que proponen no permite que ocurran en cualquier momento. De hecho, un conductor sólo puede cambiar de camión luego de haber tomado su descanso diario en una estación de relevo, y un conductor es forzado a visitar una de ellas tras transcurrir cierta cantidad de tiempo desde su última visita. Los autores siguen una descomposición del problema en dos etapas: primero, tratan con los vehículos, y posteriormente con los conductores. Proponen para su resolución una LNS que sirve para ambas etapas del problema, la cual es evaluada en instancias que provienen de una importante empresa de transporte alemana con 2800 pedidos, horizonte de planificación de 6 días, 1975 localidades, 1645 camiones y conductores, 43 depósitos, y 157 estaciones de relevo. Por desgracia, las pruebas computacionales no son concluyentes y no logran superar los resultados obtenidos al usar una correspondencia camión-conductor fija (cuando los conductores no pueden cambiar de camión).

[Lam et al., 2015] consideran una aplicación en la logística militar y humanitaria, donde las tripulaciones pueden cambiar de vehículo y también pueden trasladarse viajando como pasajeros. Los cambios de vehículo-tripulación pueden ocurrir en cualquier localidad de cliente, pero el modelo propuesto no permite que ocurran en cualquier momento, sino sólo cuando se recolecta o se entrega un pedido. Las tripulaciones pueden trabajar a lo sumo un turno, el cual tiene una duración máxima, y no se contemplan regulaciones de descanso dentro de los turnos. Los autores proponen un modelo de *Constraint Programming* que trata las dos etapas del problema al mismo tiempo, redimiendo así las limitaciones del abordaje secuencial, y el cuál es resuelto usando una LNS. Además, proponen a fines comparativos una formulación de PLE y una heurística de dos etapas, donde el PDPTW de la primera etapa se resuelve con una LNS y la planificación de las tripulaciones se resuelve

con una heurística de generación de columnas sobre un modelo de cubrimiento de conjuntos (es decir, una vez generadas las columnas, la formulación se resuelve con un *solver* de programación lineal entera). Los resultados computacionales permiten a los autores concluir que los intercambios vehículo-tripulación son esenciales para la obtención de soluciones de alta calidad, basados en un conjunto de instancias aleatorias con hasta 96 pedidos y 6 localidades.

[Domínguez-Martín et al., 2018] investigan un VRP que involucra dos depósitos. Las rutas de los vehículos deben empezar y terminar en depósitos diferentes. Por el contrario, las rutas de los conductores deben retornar a su depósito base, y para ello, los conductores deben cambiar de camión (por alguno que se dirija a su depósito base). Los intercambios solo pueden ocurrir en algunas localidades de cliente, las cuales son las únicas que pueden ser visitadas por más de un vehículo a la vez. Los conductores pueden realizar a lo sumo una ruta, cuya duración (que es el tiempo de conducción más el tiempo de viaje como pasajero) no puede exceder un tiempo máximo, y no se contemplan otras regulaciones de descanso durante la ruta. El problema es modelado con una única formulación de PLE, la cual presenta algunas limitaciones, por ejemplo: las localidades de cliente no pueden ser visitadas más de una vez por un mismo camión o conductor. Los autores proponen algunas familias de desigualdades válidas y, mediante el desarrollo de rutinas de separación para las mismas, implementaron un algoritmo B&C capaz de resolver instancias con hasta 30 clientes.

[Mendes and Iori, 2020] desarrollaron un sistema de soporte para una distribuidora farmacéutica italiana, donde los intercambios camión-conductor se permiten únicamente en un depósito central. Los autores siguen una descomposición en dos etapas: primero resuelven un VRP mediante una metaheurística *Iterated Local Search* (ILS) que determina rutas de vehículo con una duración máxima y que comienzan y finalizan en el depósito central, y posteriormente resuelven un modelo de PLE para asignar camiones y conductores a las rutas previas. Los métodos propuestos son evaluados sobre instancias reales con 187 clientes y un horizonte de planificación de una semana.

Como se ha mencionado anteriormente, una diferencia fundamental en los trabajos citados es si permiten intercambios de camión-conductor durante el horizonte de planificación. Pero existen también otras características que pueden alterar, en ma-

yor o menor medida, la naturaleza original del VRP, por ejemplo: si los pedidos son de recolección o entrega o ambos, si los pedidos tienen ventanas de tiempo, si las ventanas de tiempo abren una única vez o múltiples veces, el número de depósitos disponibles, si la flota es homogénea o heterogénea, las regulaciones de los conductores, cuándo y dónde pueden descansar los conductores, cuándo y dónde pueden ocurrir intercambios camión-conductor, los objetivos considerados, entre muchos otros. Todos estos requerimientos están motivados por aplicaciones del mundo real y la combinación de ellos genera diversas variantes de VRPs, usualmente conocidas como *Rich VRPs* (ver [Lahyani et al., 2015]).

En el Cuadro 1.3.1, se listan las principales características de los trabajos citados que están mayormente relacionados con esta investigación. De hecho, no hemos encontrado en la literatura ningún antecedente que permita intercambios de camión-conductor completamente libres, es decir, que puedan ocurrir en cualquier localidad (haya o no un cliente para visitar) y en cualquier momento (antes, durante o después de transportar un pedido). Más allá de las diferencias, cuando se trata de enfoques heurísticos, una descomposición en etapas parece ser una estrategia recurrente para enfrentar estos problemas.

Un área de investigación un poco más alejada, pero en la cual también intervienen fuertes restricciones de sincronización, es la prestación domiciliaria de servicios de salud, donde profesionales de la salud visitan a los pacientes en sus casas para suministrar tratamientos médicos [Fikar and Hirsch, 2017]. En [Eveborn et al., 2006], algunos pacientes necesitan ser visitados por más de un miembro del personal al mismo tiempo, por ejemplo, para mover pacientes con pérdida de movilidad. En [Nasir and Kuo, 2020], un vehículo debe llegar a la casa del paciente durante la visita de un enfermo para asegurar la entrega/recolección de un artículo en presencia de un personal de la salud, por ejemplo, una muestra de sangre o un desecho patológico. En estos problemas, el área de servicio suele ser mucho más limitada, por ejemplo un barrio o una ciudad, se deben proveer diferentes tipos de servicios, en particular un paciente puede necesitar más de un tipo de servicio, y cada miembro del personal tiene diferentes habilidades que limitan los servicios que puede proveer.

		[Savelsbergh and Sol, 1998]	[Xu et al., 2003]	[Goel, 2009]	[Kok et al., 2010]	[Prescott-Gagnon et al., 2010]	[Rancourt et al., 2013]	[Drexl et al., 2013]	[Goel and Vidal, 2014]	[Lam et al., 2015]	[Goel and Irnich, 2017]	[Domínguez-Martín et al., 2018]	[Koç et al., 2018]	[Goel et al., 2021]	[Mendes and Iori, 2020]	[Tilk and Goel, 2020]	Tesis actual
Problema	VRP			✓	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	
	PDP	✓	✓					✓		✓							✓
	Scheduling	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Flota de vehículos	Homogénea			✓	✓	✓			✓	✓	✓	✓	✓	✓		✓	✓
	Heterogénea	✓	✓				✓	✓							✓		
Horizonte de planificación	Simple									✓		✓					
	Múltiple	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓
Ventanas de tiempo	Ninguna											✓					
	Simples	✓		✓	✓	✓		✓	✓	✓	✓			✓	✓	✓	
	Múltiples		✓				✓		✓				✓				✓
Depósitos	Ninguno		✓														✓
	Único			✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	
	Múltiples	✓						✓				✓					
Tripulaciones	Individuales	✓	✓	✓	✓	✓		✓	✓		✓		✓	✓	✓	✓	
	Equipos													✓			
	Variables									✓		✓					✓
Regulación laboral	EEUU		✓				✓		✓		✓		✓			✓	
	UE			✓	✓	✓		✓	✓		✓			✓		✓	
	Otra	✓							✓	✓		✓			✓		✓
Lugar de relevo de conductores	Ninguno	✓	✓	✓	✓	✓	✓		✓		✓		✓	✓		✓	
	Depósitos							✓							✓		
	Localidades									✓		✓					✓
Costos de ruteo	Fijo	✓	✓					✓		✓							
	Distancia	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
	Tiempo		✓				✓							✓			
	Demora																✓
	Otro														✓		
	Ninguno											✓					
Costos de scheduling	Fijo							✓		✓					✓		
	Distancia									✓		✓					
	Tiempo		✓		✓		✓						✓	✓	✓	✓	
	Taxi							✓									✓
	Otro												✓		✓		
	Ninguno	✓		✓		✓			✓		✓						
Abordaje	Heurístico	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓		✓
	Exacto	✓								✓	✓	✓				✓	
Instancias	Benchmark			✓	✓	✓	✓		✓		✓		✓			✓	
	Generadas	✓	✓							✓		✓		✓			✓
	Reales	✓					✓	✓							✓		

Cuadro 1.3.1.: Resumen de los trabajos citados.

1.4. Objetivos generales y específicos

Objetivo general

El objetivo general de esta tesis es el estudio del problema de ruteo de vehículos y planificación de tripulaciones simultáneos, con el propósito de desarrollar métodos de resolución efectivos que permitan reducir los costos operativos y optimizar la asignación de recursos en empresas de transporte y logística.

Objetivos específicos

- Realizar una exhaustiva revisión bibliográfica acerca del estado del arte en el área, identificando las metodologías y los algoritmos más populares, así como también sus alcances y limitaciones.
- Comprender la naturaleza del problema e identificar los requisitos operativos de la empresa, con el objetivo de definir escenarios de prueba realistas y desafiantes para evaluar los métodos propuestos.
- Diseñar modelos matemáticos que integren el ruteo de vehículos con la planificación de tripulaciones, considerando la demanda de pedidos en el tiempo, la disponibilidad de camiones y conductores, la legislación laboral y demás restricciones operativas de la empresa.
- Proponer métodos heurísticos y exactos para resolver el problema planteado, con el objetivo de que los primeros pueden dar buenas soluciones en escenarios reales con gran demanda de pedidos y que los segundos puedan dar soluciones óptimas en escenarios más limitados.
- Implementar los métodos propuestos y evaluarlos sobre los diferentes escenarios de prueba para validar su rendimiento.
- Comparar cuantitativamente y discutir críticamente los resultados obtenidos en términos de su eficiencia, calidad de las soluciones encontradas y escalabilidad.

- Realizar un análisis de sensibilidad para identificar la robustez de los métodos ante diversos cambios en los parámetros que definen al problema.

1.5. Estructura de la tesis

La tesis se encuentra divida en dos partes principales, tituladas *enfoque secuencial* y *enfoque simultáneo*.

En la primera, los dos subproblemas en que se descompone el problema en estudio, ruteo de camiones y planificación de tripulaciones, se resuelven secuencialmente uno después del otro. En el Capítulo 2, se detalla la descomposición del problema en etapas y se proponen modelos matemáticos basados en PLE para cada una de ellas. En el Capítulo 3, se describen los algoritmos heurísticos desarrollados para resolver los subproblemas de cada etapa. En el Capítulo 4, se exhiben los resultados alcanzados por los algoritmos desarrollados en pruebas computacionales sobre instancias de diferente tamaño. En el Capítulo 5, se presentan las conclusiones correspondientes a la primera parte del trabajo.

En la segunda, los dos subproblemas se resuelven simultáneamente en una misma y única etapa. En el Capítulo 6, se proponen diversas formulaciones de PLE que modelan el problema. En el Capítulo 7, se estudian con mayor profundidad los modelos anteriores y se proponen desigualdades válidas. En el Capítulo 8, se desarrollan algoritmos exactos que incorporan los estudios realizados en los capítulos anteriores. En el Capítulo 9, se presentan conclusiones de la segunda parte, así como también generales a todo el trabajo realizado, y se mencionan posibles trabajos futuros.

En los apéndices finales se definen algunos conceptos preliminares necesarios para la comprensión de la tesis, que incluyen nociones de Teoría de Grafos (Apéndice 1), Optimización Combinatoria y Complejidad Computacional (Apéndice 2), Programación Lineal Entera (Apéndice 3), Heurísticas y Metaheurísticas (Apéndice 4) y Optimización Multiobjetivo (Apéndice 5).

Parte I.

Enfoque secuencial

2. Modelado con Programación Lineal Entera en dos etapas

En esta primera parte de la tesis, se trabaja con una descomposición secuencial del problema en dos subproblemas más simples y manejables: en una primera etapa se construyen rutas para los camiones y en una segunda etapa se asignan los conductores sobre las rutas de camiones construidas previamente. Es evidente que la descomposición simplifica el abordaje, puesto que los camiones y los conductores no son optimizados al mismo tiempo, permitiendo así dar soluciones a grandes instancias cuyo abordaje sería inviable de forma simultánea. Como es de esperarse, también emerge una dependencia entre ambas etapas, donde las decisiones tomadas al principio pueden restringir seriamente el espacio de búsqueda, comprometiendo así la calidad de las planificaciones encontradas posteriormente e incluso la factibilidad de las instancias.

El primero de los subproblemas involucra construir rutas para los camiones ignorando las restricciones asociadas a los conductores, de formal tal que todos los pedidos sean cumplidos dentro de sus ventanas de tiempo y minimizando las penalidades por las entregas tardías y los costos incurridos por los kilómetros recorridos. Al finalizar esta etapa, las rutas resultantes se subdividen en tareas, que son la mínima unidad de trabajo que debe ser realizada por una misma tripulación.

El segundo de los subproblemas involucra asignar tripulantes a cada una de las tareas definidas en la etapa anterior, de forma tal que se respeten las regulaciones laborales y minimizando los costos asociados al traslado en taxi de los conductores. Para una mayor flexibilidad en la asignación, en esta etapa también se permite modificar algunas decisiones tomadas en la primera, particularmente es posible adelantar y/o demorar en determinadas circunstancias el horario programado para las tareas.

En este capítulo, se modelan ambos subproblemas mediante PLE. Para esto, se recurre a digrafos con estructura que codifican rutas de camiones y de conductores como caminos dirigidos, donde los nodos representan pedidos de transporte y los arcos establecen precedencias.

2.1. Etapa I – Ruteo de camiones

El subproblema abordado en la primera etapa es una variante de un PDPTW. Involucra construir rutas de camiones que cumplan con todos los pedidos, respetando las ventanas de tiempo, y minimizando la penalidad por los días de demora en las entregas y el costo por distancia recorrida en camión. Ninguna regulación sobre los conductores es considerada durante este proceso, es decir, los camiones pueden pensarse momentáneamente como autónomos.

En lo que sigue, se propone su modelado como un problema de PLE, inspirado en el trabajo de [Dumas et al., 1991]. Para estructurar los datos, se recurre primeramente a un digrafo auxiliar que permite representar la secuencia de pedidos realizados por un camión como un camino dirigido.

2.1.1. Digrafo de secuenciación de pedidos

Se define un digrafo $G_1 = (N, E)$. El conjunto de nodos N contiene un nodo n_l para cada $l \in L_V$, un nodo n_p para cada $p \in P$ y un nodo sumidero n_s. El conjunto de arcos E está constituido por los siguientes:

- (n_l, n_p) para cada $l \in L_V$ y $p \in P$, representando que el pedido p es el primero de la ruta de un camión que inicia en la localidad l.
- (n_{p_1}, n_{p_2}) para cada $p_1, p_2 \in P$ con $p_1 \neq p_2$, representando que el pedido p_2 sigue al pedido p_1 en la ruta.
- (n_p, n_s) para cada $p \in P$, representando que el pedido p es el último de la ruta.

Para ilustrar esta construcción, se presenta el siguiente ejemplo.

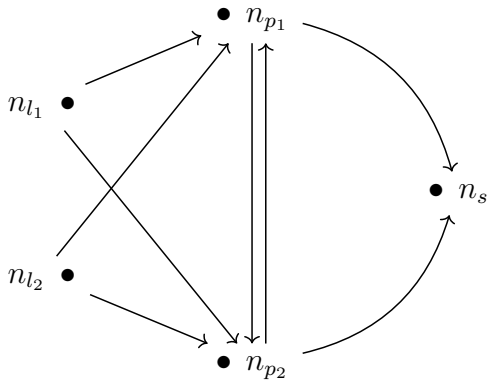


Figura 2.1.1.: Digrafo de secuenciación de pedidos para el Ejemplo 1.2.1.

Ejemplo 2.1.1. En la Figura 2.1.1, se observa el digrafo de secuenciación de pedidos asociado al Ejemplo 1.2.1.

2.1.2. Variables y restricciones

Para modelar este problema usando PLE, se requiere de una variable $x_e \in \{0, 1\}$ para cada $e \in E$, tal que $x_e = 1$ si y sólo si un camión pasa por el arco e. La razón por la cual estas variables se consideran binarias es porque cada pedido debe ser realizado por exactamente un camión, luego cada arco es recorrido por a lo sumo uno de ellos. Estas variables deben estar sujetas a las siguientes restricciones.

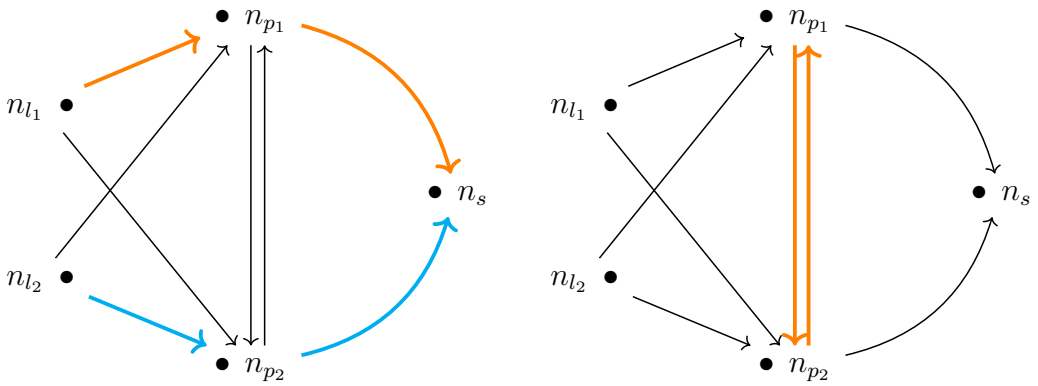
- La cantidad de camiones usados no puede exceder a los disponibles en cada localidad inicial l:

$$\sum_{e \in \Gamma^+(n_l)} x_e \leq |\{v \in V : l_v = l\}| \qquad \forall l \in L_V$$

  Observar que esto permite que algunos camiones no sean utilizados para ejecutar pedidos.

- El flujo de camiones se conserva en cada nodo asociado a un pedido p:

$$\sum_{e \in \Gamma^-(n_p)} x_e = \sum_{e \in \Gamma^+(n_p)} x_e \qquad \forall p \in P$$



(a) Solución correcta. (b) Solución incorrecta.

Figura 2.1.2.: Posibles soluciones para el Ejemplo 1.2.1.

- Cada pedido p es realizado por exactamente un camión:

$$\sum_{e \in \Gamma^-(n_p)} x_e = 1 \qquad \forall p \in P$$

$$\sum_{e \in \Gamma^+(n_p)} x_e = 1 \qquad \forall p \in P$$

  Por las restricciones de conservación de flujo, una de estas dos familias de restricciones queda implicada y puede omitirse de la formulación.

Sin embargo, esta propuesta no es suficiente para modelar el subproblema y sus limitaciones se exponen en el siguiente ejemplo.

Ejemplo 2.1.2. En la Figura 2.1.2a, el camino dirigido naranja representa la ruta del camión v_1 del Ejemplo 1.2.1 y el camino dirigido cyan la ruta de v_2. Es fácil ver que estas rutas verifican las restricciones anteriores considerando $x_e = 1$ para los arcos e coloreados y $x_e = 0$ para los demás. Sin embargo, en la Figura 2.1.2b, se muestra una solución que también las verifica, pero que no se corresponde con una ruta de camión.

Aún quedan dos cuestiones por resolver en este modelo. Por un lado, es necesario prohibir ciclos, y por el otro, resta programar un día y una hora para iniciar la recolección y la entrega de cada pedido. Afortunadamente, ambas cuestiones se pueden resolver introduciendo nuevas variables y restricciones que incorporen una dimensión temporal. Luego, se consideran las siguientes variables:

- $d_p^{\mathrm{R}}, d_p^{\mathrm{E}} \in \mathbb{Z}$ para cada $p \in P$, que modelan días. El rango de estas variables depende de si la ventana de tiempo comienza y finaliza en el mismo día. Más precisamente, $d_p^{\mathrm{R}} \in \{dia_p^{\mathrm{R}}, \ldots, H-1\}$ si $a_p^{\mathrm{R}} < b_p^{\mathrm{R}}$, y $d_p^{\mathrm{R}} \in \{dia_p^{\mathrm{R}} - 1, \ldots, H-1\}$ si $a_p^{\mathrm{R}} > b_p^{\mathrm{R}}$, análogamente para d_p^{E}.
- $h_p^{\mathrm{R}}, h_p^{\mathrm{E}} \in \mathbb{N}_0$ para cada $p \in P$, que modelan instantes de tiempo. El rango de estas variables es $h_p^{\mathrm{R}} \in \{a_p^{\mathrm{R}}, \ldots, b_p^{\mathrm{R}}\}$ si $a_p^{\mathrm{R}} < b_p^{\mathrm{R}}$, y $h_p^{\mathrm{R}} \in \{a_p^{\mathrm{R}}, \ldots, I + b_p^{\mathrm{R}}\}$ si $a_p^{\mathrm{R}} > b_p^{\mathrm{R}}$, análogamente para h_p^{E}.

La idea es que $I.d_p^{\mathrm{R}} + h_p^{\mathrm{R}}$ represente el instante de tiempo en que comienza la recolección de p y $I.d_p^{\mathrm{E}} + h_p^{\mathrm{E}}$ su entrega. Antes de detallar las restricciones involucradas, véase el siguiente ejemplo.

Ejemplo 2.1.3. Considerar las rutas propuestas en el Ejemplo 1.2.1, con $H = 1$ e $I = 8$. Dado que el pedido p_2 se ejecuta en el instante de tiempo 3 del primer día, las variables deben asumir los valores $d_{p_2}^{\mathrm{R}} = 0 \in \{0\}$ y $h_{p_2}^{\mathrm{R}} = 3 \in \{3, 4, 5\}$. Por el contrario, para el pedido p_1 que se ejecuta en el instante de tiempo 2 del primer día, las variables deben asumir los valores $d_{p_1}^{\mathrm{E}} = -1 \in \{-1, 0\}$ y $h_{p_1}^{\mathrm{E}} = 10 \in \{6, \ldots, 10\}$.

Las variables deben estar sujetas a las siguientes restricciones:

- Si un pedido p es el primero de la ruta de un camión que inicia en la localidad l, entonces el instante de tiempo en el que inicia la recolección de p debe ser mayor o igual al tiempo de viaje desde l a la localidad de recolección de p:

$$I.d_p^{\mathrm{R}} + h_p^{\mathrm{R}} \geq \mathrm{tiempoViajeCamion}(l, l_p^{\mathrm{R}}).x_{(n_l, n_p)} \qquad \forall l \in L_V,\ p \in P$$

- El instante de tiempo en el que inicia la entrega del pedido p es posterior al instante en que se recolecta, sumado al tiempo de servicio de recolección y al tiempo de viaje entre las localidades de recolección y de entrega:

$$I.d_p^{\mathrm{E}} + h_p^{\mathrm{E}} \geq I.d_p^{\mathrm{R}} + h_p^{\mathrm{R}} + s_p^{\mathrm{R}} + \mathrm{tiempoViajeCamion}(l_p^{\mathrm{R}}, l_p^{\mathrm{E}}) \qquad \forall p \in P$$

- Si un camión realiza el pedido p_2 luego de p_1, entonces el instante de tiempo en que inicia la recolección de p_2 debe ser posterior al instante en que se

entrega p_1, sumado al tiempo de servicio de entrega de p_1 y al tiempo de viaje entre las localidades de entrega de p_1 y de recolección de p_2:

$$I.d^{\mathrm{R}}_{p_2} + h^{\mathrm{R}}_{p_2} + M.(1 - x_{(n_{p_1}, n_{p_2})}) \geq$$
$$I.d^{\mathrm{E}}_{p_1} + h^{\mathrm{E}}_{p_1} + s^{\mathrm{E}}_{p_1} + \mathrm{tiempoViajeCamion}(l^{\mathrm{E}}_{p_1}, l^{\mathrm{R}}_{p_2}) \quad \forall p_1, p_2 \in P,\ p_1 \neq p_2$$

donde M es una constante *big M* con un valor muy grande, por ejemplo $M = I.H$.

- La recolección y la entrega de un pedido p inician luego del primer día permitido por sus ventanas de tiempo:

$$I.d^{\mathrm{R}}_p + h^{\mathrm{R}}_p \geq I.dia^{\mathrm{R}}_p \qquad \forall p \in P \text{ con } a^{\mathrm{R}}_p > b^{\mathrm{R}}_p$$
$$I.d^{\mathrm{E}}_p + h^{\mathrm{E}}_p \geq I.dia^{\mathrm{E}}_p \qquad \forall p \in P \text{ con } a^{\mathrm{E}}_p > b^{\mathrm{E}}_p$$

  Las mismas son necesarias dado que, por ejemplo, para un pedido p cuya ventana de tiempo de recolección comienza y finaliza en días distintos, los rangos de las variables permiten la asignación $d^{\mathrm{R}}_p = dia^{\mathrm{R}}_p - 1$ y $a^{\mathrm{R}}_p \leq h^{\mathrm{R}}_p < I$. Es decir, la recolección se llevaría a cabo antes del primer día permitido.

- La entrega de todo pedido p finaliza antes del último instante del horizonte de planificación:

$$I.d^{\mathrm{E}}_p + h^{\mathrm{E}}_p + s^{\mathrm{E}}_p \leq I.H \qquad \forall p \in P$$

2.1.3. Funciones objetivo

En esta etapa, se busca minimizar los objetivos relacionados con los camiones, esto es, la penalidad por los días de demora en las entregas y el costo por distancia recorrida en camión. Cada uno de ellos requerirá de una estrategia diferente para su modelado.

- *Minimizar penalidad por demora en las entregas:*

  Dado un pedido p, la variable d^{E}_p guarda información sobre el día en que un camión realiza efectivamente la entrega de p. Sin embargo, hay un problema cuando la ventana de tiempo de entrega abre y cierra en días distintos, pues

el valor de d_p^{E} no siempre coincide con el día de entrega. Más precisamente, d_p^{E} almacena el día de entrega cuando $h_p^{\mathrm{E}} \in \{a_p^{\mathrm{E}}, \ldots, I-1\}$ y el día anterior cuando $h_p^{\mathrm{E}} \in \{I, \ldots, I + b_p^{\mathrm{E}}\}$.

Para solucionarlo, se introduce una variable binaria z_p para cada $p \in P$ con $a_p^{\mathrm{E}} > b_p^{\mathrm{E}}$, tal que $z_p = 1$ si $h_p^{\mathrm{E}} \geq I$. La activación de estas variables está controlada por las siguientes restricciones:

$$h_p^{\mathrm{E}} - I.z_p \leq I - 1 \qquad \forall p \in P \text{ con } a_p^{\mathrm{E}} > b_p^{\mathrm{E}}.$$

Notar que si h_p^{E} es menor que I, no se impone ninguna restricción al valor de z_p, y en este caso, la función objetivo evitará que la variable se active innecesariamente.

Como resultado, d_p^{E} representa el día en que se entrega el pedido p cuando $a_p^{\mathrm{E}} < b_p^{\mathrm{E}}$ y $d_p^{\mathrm{E}} + z_p$ cuando $a_p^{\mathrm{E}} > b_p^{\mathrm{E}}$. Con esta distinción, la función objetivo se puede expresar de la siguiente forma:

$$\text{mín} \sum_{p \in P} w_p.(d_p^{\mathrm{E}} - dia_p^{\mathrm{E}}) + \sum_{\substack{p \in P: \\ a_p^{\mathrm{E}} > b_p^{\mathrm{E}}}} w_p.z_p$$

Es decir, la suma de las penalidades multiplicadas por los días de demora en las entregas.

- *Minimizar costo por distancia recorrida en camión:*

Para modelar este objetivo, se incorporan pesos a los arcos de G_1. El peso de cada arco dependerá del costo de viaje en camión que requiere la transición representada por el mismo. Más precisamente, se define la función costoCamion, que toma un arco e de G_1 y retorna su costo, como:

$$\text{costoCamion}(e) =$$
$$\begin{cases} \text{costoViajeCamion}(l, l_p^{\mathrm{R}}) + \text{costoViajeCamion}(l_p^{\mathrm{R}}, l_p^{\mathrm{E}}) & \text{si } e = (n_l, n_p), \\ \text{costoViajeCamion}(l_{p_1}^{\mathrm{E}}, l_{p_2}^{\mathrm{R}}) + \text{costoViajeCamion}(l_{p_2}^{\mathrm{R}}, l_{p_2}^{\mathrm{E}}) & \text{si } e = (n_{p_1}, n_{p_2}), \\ 0 & \text{caso contrario.} \end{cases}$$

Es decir, el costo de (n_l, n_p) es el de viajar desde la localidad n_l a la localidad de recolección del pedido p, más el costo de viajar entre las localidades de recolección y de entrega de p. El costo de (n_{p_1}, n_{p_2}) es el de viajar de la localidad de entrega del pedido p_1 a la de recolección del pedido p_2, más el costo de viajar entre las localidades de recolección y de entrega de p_2. Y finalmente, el costo es 0 para los demás arcos, ya que no se requiere que los camiones regresen a su localidad inicial al finalizar su ruta.

Luego, este objetivo se puede expresar como la suma de los pesos de los arcos recorridos por los camiones:

$$\text{mín} \sum_{e \in E} \text{costoCamion}(e).x_e$$

2.1.4. Formulación E1

A continuación se detalla la formulación de PLE que resulta de unir todos los elementos presentados.

$$\text{(E1)} \quad \text{mín} \sum_{p \in P} w_p.(d_p^{\text{E}} - dia_p^{\text{E}}) + \sum_{\substack{p \in P: \\ a_p^{\text{E}} > b_p^{\text{E}}}} w_p.z_p \qquad (f_{\text{ENTR}})$$

$$\text{mín} \sum_{e \in E} \text{costoCamion}(e).x_e \qquad (f_{\text{VIAJ}})$$

$$\text{Sujeto a:} \sum_{e \in \Gamma^+(n_l)} x_e \leq |\{v \in V : l_v = l\}| \qquad \forall l \in L_V \qquad (2.1)$$

$$\sum_{e \in \Gamma^-(n_p)} x_e = \sum_{e \in \Gamma^+(n_p)} x_e \qquad \forall p \in P \qquad (2.2)$$

$$\sum_{e \in \Gamma^-(n_p)} x_e = 1 \qquad \forall p \in P \qquad (2.3)$$

$$I.d_p^{\text{R}} + h_p^{\text{R}} \geq \text{tVC}(l, l_p^{\text{R}}).x_{(n_l, n_p)} \qquad \forall l \in L_V,\ p \in P \qquad (2.4)$$

$$I.d_p^{\text{E}} + h_p^{\text{E}} \geq I.d_p^{\text{R}} + h_p^{\text{R}} + s_p^{\text{R}} + \text{tVC}(l_p^{\text{R}}, l_p^{\text{E}}) \qquad \forall p \in P \qquad (2.5)$$

$$I.d_{p_2}^{\text{R}} + h_{p_2}^{\text{R}} + M.(1 - x_{(n_{p_1}, n_{p_2})}) \geq I.d_{p_1}^{\text{E}} + h_{p_1}^{\text{E}} + s_{p_1}^{\text{E}} + \text{tVC}(l_{p_1}^{\text{E}}, l_{p_2}^{\text{R}}) \qquad \forall p_1, p_2 \in P,\ p_1 \neq p_2 \qquad (2.6)$$

$$I.d_p^{\text{R}} + h_p^{\text{R}} \geq I.dia_p^{\text{R}} \qquad \forall p \in P \text{ con } a_p^{\text{R}} > b_p^{\text{R}} \qquad (2.7)$$

$$I.d_p^{\text{E}} + h_p^{\text{E}} \geq I.dia_p^{\text{E}} \qquad \forall p \in P \text{ con } a_p^{\text{E}} > b_p^{\text{E}} \qquad (2.8)$$

$$I.d_p^{\mathrm{E}} + h_p^{\mathrm{E}} + s_p^{\mathrm{E}} \leq I.H \qquad \forall p \in P \qquad (2.9)$$

$$h_p^{\mathrm{E}} - I.z_p \leq I - 1 \qquad \forall p \in P \text{ con } a_p^{\mathrm{E}} > b_p^{\mathrm{E}} \qquad (2.10)$$

$$x_e \in \{0,1\} \qquad \forall e \in E \qquad (2.11)$$

$$d_p^y \in \{dia_p^y, \ldots, H-1\} \qquad \forall y \in \{\mathrm{R},\mathrm{E}\},\ p \in P \text{ con } a_p^y < b_p^y \qquad (2.12)$$

$$d_p^y \in \{dia_p^y - 1, \ldots, H-1\} \qquad \forall y \in \{\mathrm{R},\mathrm{E}\},\ p \in P \text{ con } a_p^y > b_p^y \qquad (2.13)$$

$$h_p^y \in \{a_p^y, \ldots, b_p^y\} \qquad \forall y \in \{\mathrm{R},\mathrm{E}\},\ p \in P \text{ con } a_p^y < b_p^y \qquad (2.14)$$

$$h_p^y \in \{a_p^y, \ldots, I + b_p^y\} \qquad \forall y \in \{\mathrm{R},\mathrm{E}\},\ p \in P \text{ con } a_p^y > b_p^y \qquad (2.15)$$

$$z_p \in \{0,1\} \qquad \forall p \in P \text{ con } a_p^{\mathrm{E}} > b_p^{\mathrm{E}} \qquad (2.16)$$

Dada una solución factible de esta formulación y una localidad inicial $l \in L_V$, todo (n_l, n_s)-camino dirigido de G_1 cuyos arcos e tengan $x_e = 1$ representa la ruta de un camión que inicia en l. El resto de las variables indican el instante de tiempo en que se realiza la recolección y la entrega de cada pedido. Sin embargo, es importante notar que la formulación no fija el horario exacto en que los camiones viajan, pero se tiene la garantía de que existe una asignación de horarios factible para ellos, compatible con el horario fijado para las recolecciones y las entregas.

2.1.5. Segmentación de rutas en tareas

Antes de comenzar la segunda etapa, las rutas de camiones se deben segmentar en *tareas*, que es la mínima unidad de trabajo que debe ser realizada íntegramente por una misma tripulación. Es decir, durante el transcurso de una tarea no es posible cambiar a los conductores, pero dos tareas consecutivas en una misma ruta pueden variar sus tripulaciones.

Siendo v un vehículo, se utiliza el siguiente procedimiento para segmentar su ruta en una secuencia ordenada de tareas.

- Cada vez que v recolecta un pedido p, se tiene una tarea de recolección cuyo lugar de origen y destino es l_p^{R} y su duración es s_p^{R}.
- Cada vez que v entrega un pedido p, se tiene una tarea de entrega cuyo lugar de origen y destino es l_p^{E} y su duración es s_p^{E}.
- Cada vez que v viaja desde una localidad l hacia otra l', se busca una secuencia de localidades $l_1, \ldots, l_{n+1}$ con $n \in \mathbb{N}$, $l_1 = l$ y $l_{n+1} = l'$, formada por

todas las localidades intermedias por las cuales pasa la ruta más de corta desde l hacia l'. Dado que en cada localidad intermedia se pueden cambiar los conductores, se tienen n tareas de viaje, donde la tarea j-ésima, con $j \in \{1, \ldots, n\}$, tiene por localidad de origen a l_j, por localidad de destino a l_{j+1} y su duración es tiempoViajeCamion(l_j, l_{j+1}).

Es útil introducir notaciones para referir a estos nuevos elementos.

- T^{R}, T^{E} y T^{V} son los conjuntos que contienen todas las tareas de recolección, de entrega y de viaje, respectivamente, involucradas en las rutas de todos los camiones.
- $T = T^{\mathrm{R}} \cup T^{\mathrm{E}} \cup T^{\mathrm{V}}$.
- Para cada $t \in T$, $\mathrm{ori}(t) \in L$ es su localidad de origen, $\mathrm{des}(t) \in L$ es su localidad de destino, $\mathrm{dur}(t) \in \mathbb{N}$ es su duración en instantes de tiempo y $\mathrm{veh}(t) \in V$ es el camión que la realiza. Para cada $t \in T^{\mathrm{R}} \cup T^{\mathrm{E}}$, $\mathrm{ped}(t) \in P$ es el pedido involucrado en esa tarea recolección o entrega.
- Se definen las siguientes relaciones binarias en T, donde una tarea t_1 está relacionada con una tarea t_2 si:
  - Son ejecutadas por el mismo camión y t_2 es posterior a t_1 (no necesariamente consecutivas):

    $T^{\rightarrow} = \{(t_1, t_2) \in T \times T : \mathrm{veh}(t_1) = \mathrm{veh}(t_2) \wedge t_2 \text{ es posterior a } t_1\}$.
  - Son ejecutadas por el mismo camión, t_2 es posterior a t_1 y son consecutivas:

    $T^{\Rightarrow} = \{(t_1, t_2) \in T^{\rightarrow} : \nexists\, t \in T \text{ tal que } (t_1, t), (t, t_2) \in T^{\rightarrow}\}$.
  - Son ejecutadas por camiones diferentes:

    $T^{\neq} = \{(t_1, t_2) \in T \times T : \mathrm{veh}(t_1) \neq \mathrm{veh}(t_2)\}$.

Para ejemplificar este procedimiento, se introduce el siguiente ejemplo.

Ejemplo 2.1.4. La segmentación de las rutas de camión de v_1 y v_2 del Ejemplo 1.2.1 generan las siguientes tareas:

Tarea t	ori(t)	des(t)	dur(t)	veh(t)	ped(t)
$t_1 \in T^{\mathrm{R}}$	l_1	l_1	1	v_1	p_1
$t_2 \in T^{\mathrm{V}}$	l_1	l_2	1	v_1	–
$t_3 \in T^{\mathrm{E}}$	l_2	l_2	1	v_1	p_1
$t_4 \in T^{\mathrm{R}}$	l_2	l_2	1	v_2	p_2
$t_5 \in T^{\mathrm{V}}$	l_2	l_1	1	v_2	–
$t_6 \in T^{\mathrm{E}}$	l_1	l_1	1	v_2	p_2

Como se ha mencionado previamente, una solución de la formulación (E1) especifica el horario en que los camiones realizan las tareas de recolección y de entrega, pero no especifica el horario de las tareas de viaje. Considerando los objetivos involucrados en la primera etapa, el horario de las tareas de viaje es indiferente y se puede elegir dentro de un rango sin afectar el valor objetivo de la solución. Esto es, si una secuencia de tareas de viaje $t_1, \ldots, t_n$ se debe realizar en un intervalo de tiempo $[a, b]$, entonces para el horario de t_1 se puede elegir un entero

$$i_1 \in [a, b - \sum_{i=1}^{n} \mathrm{dur}(t_i)],$$

y para el horario de t_j, con $1 < j \leq n$, se puede elegir un entero

$$i_j \in [i_{j-1} + \mathrm{dur}(t_{j-1}), b - \sum_{i=j}^{n} \mathrm{dur}(t_i)].$$

Por el contrario, el horario en que se programan los viajes no es indiferente para los conductores. Por ejemplo, un viaje se podría demorar o adelantar para esperar a que un conductor se desocupe o para cumplir con su descanso. Dado que la planificación de tripulaciones puede ser demasiado restrictiva cuando los horarios de las tareas están fijos, durante la segunda etapa se permite modificarlos, siempre y cuando se preserve el orden en que los camiones las realizan. Para formalizar esto, se introducen las siguientes definiciones.

Definición 2.1.5. Dada una tarea t, se define el subconjunto $\mathcal{I}_t$ de $\mathcal{I}$ con los

posibles instantes de tiempo en los cuales puede comenzar t. Formalmente,

$$\mathcal{I}_t = \begin{cases} \mathcal{I}^{\mathrm{R}}_{\mathrm{ped}(t)} & \text{si } t \in T^{\mathrm{R}} \\ \mathcal{I}^{\mathrm{E}}_{\mathrm{ped}(t)} & \text{si } t \in T^{\mathrm{E}} \\ \{i \in \mathcal{I} : i \leq I.H - \mathrm{dur}(t)\} & \text{si } t \in T^{\mathrm{V}} \end{cases}$$

Definición 2.1.6. Una *programación horaria* es una función $\delta : T \to \mathcal{I}$ que determina el instante de tiempo en el que comienza cada tarea. Debe verificar las siguientes condiciones:

1. $\delta(t) \in \mathcal{I}_t$ para todo $t \in T$.
2. $\delta(t_1) + \mathrm{dur}(t_1) \leq \delta(t_2)$ para todo $(t_1, t_2) \in T^{\Rightarrow}$. Es decir, si dos tareas t_1 y t_2 son consecutivas en una ruta de camión, entonces t_2 debe comenzar luego de que t_1 termine.

En particular, se nota con δ_0 a la programación horaria que se obtiene de la solución de la primera etapa, completando con horarios aleatorios para las tareas de viaje (de acuerdo al procedimiento mencionado anteriormente).

Para ilustrar estas definiciones, se presenta el siguiente ejemplo.

Ejemplo 2.1.7. La programación horaria δ para el Ejemplo 1.2.1 está dada por: $\delta(t_1) = 0$, $\delta(t_2) = 1$, $\delta(t_3) = 2$, $\delta(t_4) = 3$, $\delta(t_5) = 4$ y $\delta(t_6) = 5$.

Es fácil ver que para cualquier programación horaria, la solución tendrá el mismo costo respecto a la distancia de viaje en camión (dado que los camiones realizan la misma secuencia de viajes, pero programados en horarios diferentes). Sin embargo, la programación horaria puede cambiar el día en que se hacen las entregas, variando así el costo de la solución respecto a la demora en las entregas. Para preservar el costo de la solución en la segunda etapa respecto a los objetivos involucrados en la primera, de aquí en adelante, para cada tarea $t \in T^{\mathrm{E}}$, se restringe $\mathcal{I}_t$ a

$$\{i \in \mathcal{I}^{\mathrm{E}}_{\mathrm{ped}(t)} : \mathrm{dia}(i) = \mathrm{dia}(\delta_0(t))\},$$

es decir, las tareas de entrega se deben realizar el mismo día en que fueron programadas en la primera etapa.

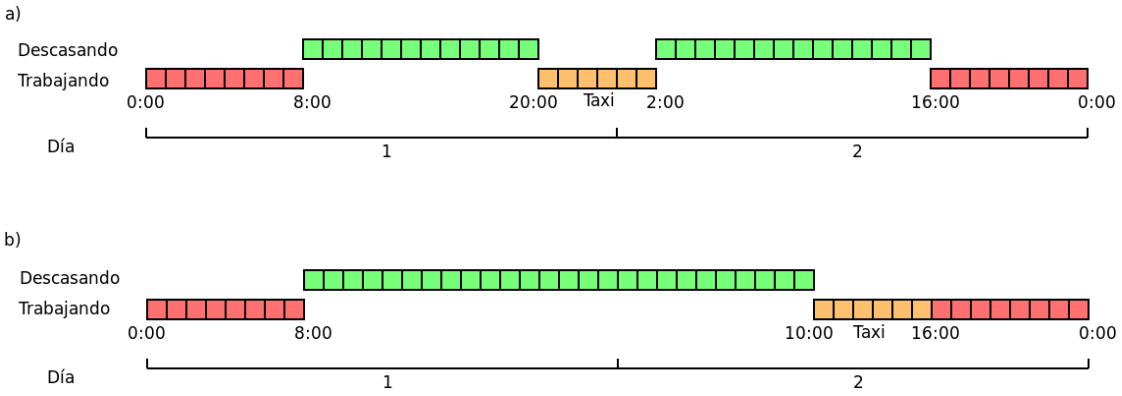


Figura 2.2.1.: Ejemplos de programación de viaje en taxi. a) factible, b) no factible.

2.2. Etapa II – Planificación de tripulaciones

El subproblema abordado en la segunda etapa involucra asignar tripulaciones, constituidas por uno o dos conductores, a cada una de las tareas definidas en la primera etapa y decidir una programación horaria para las mismas, respetando las regulaciones laborales. Además, es posible programar viajes en taxi para los conductores cuando sea necesario o conveniente, teniendo en cuenta que en esta etapa se busca minimizar el costo por la distancia de viaje recorrida por dichos transportes externos.

La programación de horarios para los viajes en taxi no es trivial, debido a que su duración contabiliza como horas trabajadas para los conductores. De hecho, puede considerarse como otro problema de optimización en sí mismo, como lo ilustra el siguiente ejemplo.

Ejemplo 2.2.1. Suponer que un conductor trabaja el primer día desde las 0:00 hasta las 8:00, el segundo día desde las 16:00 hasta las 0:00 y que necesita un viaje en taxi intermedio de 6 horas de duración. Si la partida del taxi se programa el primer día entre las 20:00 y las 22:00, entonces el conductor tiene sus 12 horas de descanso en todo intervalo de 24 horas. Uno de estos escenarios posibles se muestra en la Figura 2.2.1a). Por el contrario, si el taxi se programa el segundo día a las 10:00, entonces la solución es infactible porque el conductor trabaja 14 horas en el intervalo $[24, 48]$, ver la Figura 2.2.1b).

Con el objetivo de limitar la combinatoria generada por los taxis, en esta primera parte de la tesis, se asumirá que los mismos no se pueden interrumpir (por ejemplo, para que un conductor pueda parar a descansar en el medio) y que se programan lo más tarde posible (es decir, de acuerdo a la Figura 2.2.1b). Vale la pena mencionar que programarlos lo más temprano posible sería igual de efectivo para este propósito, por lo que ambos criterios son a priori indiferentes. La imposición de estas nuevas restricciones conduce a un problema más restringido, donde toda solución es a su vez solución del problema original, pero la inversa no es verdadera, y por ende habrá soluciones que no puedan ser encontradas por el modelo. Aún a pesar de esta limitación, creemos que es más realista que simplemente ignorar la duración de los viajes en taxi, como se plantea en [Drexl et al., 2013].

Para estructurar los datos de este problema, se recurre nuevamente a un digrafo auxiliar que permite representar la secuencia de tareas realizadas por un conductor como un camino dirigido.

2.2.1. Digrafo de secuenciación de tareas

Se define un digrafo $G_2 = (N, E)$. El conjunto de nodos N contiene un nodo n_l para cada $l \in L_C$, un nodo n_t para cada $t \in T$ y un nodo sumidero n_s. El conjunto de arcos E está constituido por los siguientes:

- (n_l, n_t) para cada $l \in L_C$ y $t \in T$, representando que la tarea t es la primera de la ruta de un conductor que inicia en la localidad l.
- (n_{t_1}, n_{t_2}) para cada $t_1, t_2 \in T^{\rightarrow}$ y (n_{t_1}, n_{t_2}) para cada $t_1, t_2 \in T^{\neq}$, representando que algún conductor realiza la tarea t_2 inmediatamente después de la tarea t_1 en su ruta (notar que estas tareas no necesariamente deben ser consecutivas en la ruta de un camión).
- (n_l, n_s) para cada $l \in L_C$, representando que la ruta de un conductor que inicia en la localidad l no realiza ninguna tarea.
- (n_t, n_s) para cada $t \in T$, representando que la tarea t es la última de la ruta de algún conductor.

A continuación, se presenta un ejemplo de esta construcción.

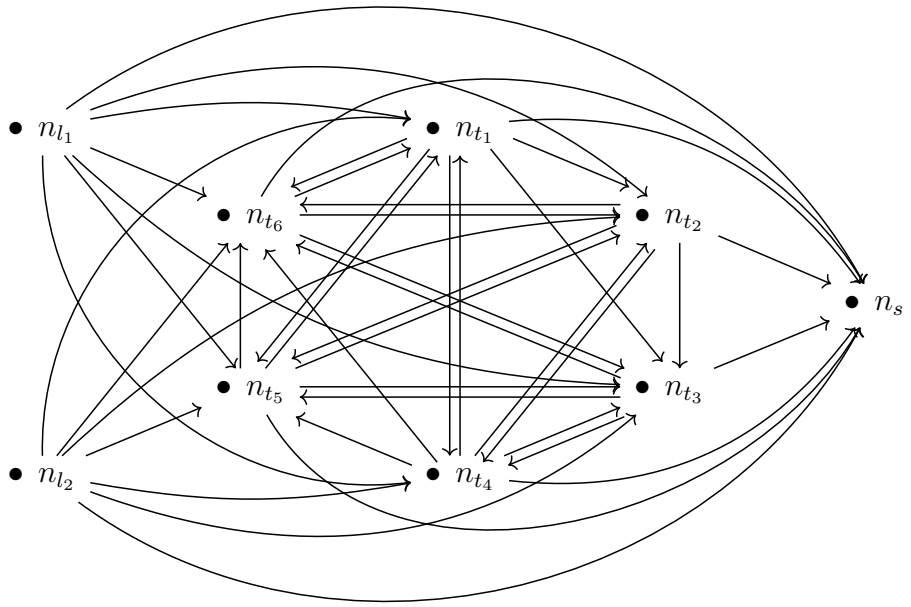


Figura 2.2.2.: Digrafo de secuenciación de tareas para el Ejemplo 1.2.1.

Ejemplo 2.2.2. En la Figura 2.2.2, se muestra el digrafo de secuenciación de tareas que resulta para el Ejemplo 1.2.1 usando la segmentación de tareas propuesta en el Ejemplo 2.1.4.

2.2.2. Variables y restricciones

Sea E_c el subconjunto de arcos que pueden ser recorridos por un conductor c, es decir, $E_c = E \setminus \{(n_l, n_t) : l \in L_C,\ l \neq l_c,\ t \in T\}$. Para modelar este problema usando PLE, se requiere de una variable $x_{ce} \in \{0, 1\}$ para cada $c \in C$ y $e \in E_c$, tal que $x_{ce} = 1$ si y sólo el conductor c pasa por el arco e. Estas variables deben estar sujetas a las siguientes restricciones:

- Cada conductor c recorre exactamente uno de los arcos que salen de su localidad inicial:

$$\sum_{e \in \Gamma^+(n_{l_c})} x_{ce} = 1 \qquad \forall c \in C$$

- El flujo de cada conductor c se conserva en cada nodo asociado a una tarea t:

$$\sum_{e \in \Gamma^-(n_t) \cap E_c} x_{ce} = \sum_{e \in \Gamma^+(n_t) \cap E_c} x_{ce} \qquad \forall c \in C,\ t \in T$$

- El número de conductores que realizan una tarea t es 1 o 2:

$$1 \leq \sum_{c \in C} \sum_{e \in \Gamma^-(n_t) \cap E_c} x_{ce} \leq 2 \qquad \forall t \in T$$
$$1 \leq \sum_{c \in C} \sum_{e \in \Gamma^+(n_t) \cap E_c} x_{ce} \leq 2 \qquad \forall t \in T$$

  Por las restricciones de conservación de flujo, una de estas dos familias de restricciones queda implicada y puede omitirse de la formulación.

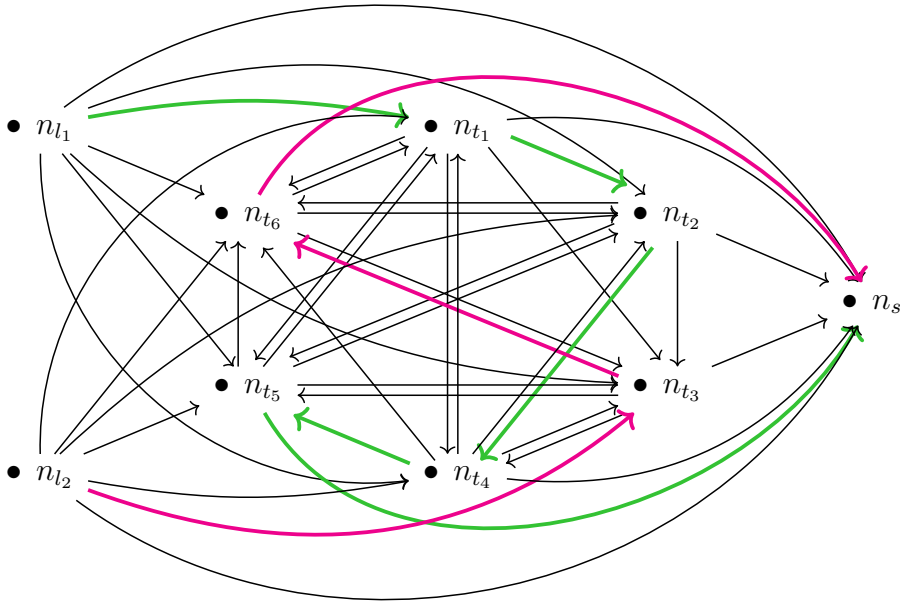
El siguiente ejemplo expone algunas limitaciones de este modelo.

Ejemplo 2.2.3. En la Figura 2.2.3a, el camino dirigido verde representa la ruta del conductor c_1 del Ejemplo 1.2.1 y el camino dirigido magenta la ruta de c_2. Es fácil ver que esta solución verifica todas las restricciones anteriores, considerando $x_{c_1 e} = 1$ para los arcos e verdes, $x_{c_2 e} = 1$ para los arcos e magenta y todas las demás variables con valor 0. Sin embargo, en la Figura 2.2.3b, se muestra una solución que también verifica las restricciones, pero que no se corresponde con rutas de conductores válidas.

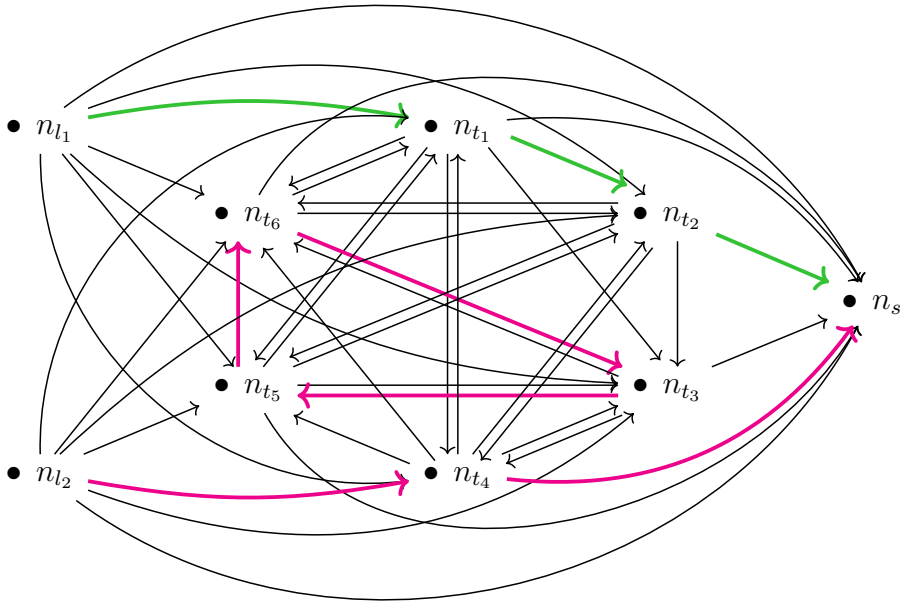
Esto motiva la incorporación de nuevas variables, tanto para prohibir ciclos, como para decidir el horario de las tareas.

- Para cada $t \in T$ e instante de tiempo $i \in \mathcal{I}_t$, se agrega una variable $y_{ti} \in \{0, 1\}$, tal que $y_{ti} = 1$ si y sólo si la tarea t comienza en el instante i.
- Para cada $e \in E$, se agrega una variable $X_e \in \{0, 1\}$, tal que $X_e = 1$ si existe $c \in C$ tal que $x_{ce} = 1$, es decir, si algún conductor pasa por el arco e.

  Notar que X_e podría estar activada aún cuando ningún conductor pase por e, sin embargo, la activación innecesaria de esta variable no afectará la factibilidad de las soluciones.



(a) Solución correcta.



(b) Solución incorrecta.

Figura 2.2.3.: Posibles soluciones para el Ejemplo 1.2.1.

Estas variables deben estar sujetas a las siguientes restricciones.

- Toda tarea t comienza en un único instante de tiempo:

$$\sum_{i \in \mathcal{I}_t} y_{ti} = 1 \qquad \forall t \in T$$

  Al imponer estas restricciones (sumado a que las variables son binarias), es posible determinar el instante de tiempo en el que comienza cada tarea t mediante la siguiente función lineal:

$$\delta(t) = \sum_{i \in \mathcal{I}_t} i.y_{ti}.$$

  Para simplificar la redacción del modelo, haremos uso de la mencionada notación para las próximas restricciones.

- La variable X_e se activa cuando algún conductor c recorre el arco e:

$$\sum_{c \in C} x_{ce} \leq |C|.X_e \qquad \forall e \in E$$

- Si una tarea t es la primera de la ruta de un conductor que inicia en la localidad l, entonces el horario de t se debe programar considerando el tiempo de viaje en taxi desde l a la localidad origen de t:

$$\delta(t) \geq \text{tiempoViajeTaxi}(l, \text{ori}(t)).X_{(n_l, n_t)} \qquad \forall l \in L_C,\ t \in T$$

- Si dos tareas t_1 y t_2 son consecutivas en una ruta de camión, entonces el horario de t_2 se debe programar luego de que t_1 finalice; más aún, si un mismo conductor realiza t_2 luego de t_1, entonces la programación debe tener en cuenta el tiempo de viaje en taxi desde la localidad de destino de t_1 a la de origen de t_2:

$$\delta(t_2) \geq \delta(t_1) + \text{dur}(t_1) + \text{tiempoViajeTaxi}(\text{des}(t_1), \text{ori}(t_2)).X_{(n_{t_1}, n_{t_2})} \qquad \forall (t_1, t_2) \in T^{\Rightarrow}$$

Notar que estas restricciones garantizan que la programación horaria satisface la condición 2 de la Definición 2.1.6.

- Si un mismo conductor realiza una tarea t_2 luego de una tarea t_1 y las tareas son realizadas por camiones diferentes, entonces el horario de t_2 se debe programar luego de que t_1 finalice y considerando el tiempo de viaje en taxi desde la localidad de destino de t_1 a la de origen de t_2; si ningún conductor realiza consecutivamente estas tareas, entonces la programación horaria de t_2 debe ser independiente de la de t_1:

$$\delta(t_2) \geq \delta(t_1) + \mathrm{dur}(t_1) \; - M.(1 - X_{(n_{t_1}, n_{t_2})}) + \\ \mathrm{tiempoViajeTaxi}(\mathrm{des}(t_1), \mathrm{ori}(t_2)) \qquad \forall (t_1, t_2) \in T^{\neq}$$

Resta introducir restricciones que modelen las regulaciones laborales de los conductores. Para ello, es necesario introducir una nueva variable $w_{ci} \in \{0, 1\}$ para cada $c \in C$ e $i \in \{0, \ldots, I.H - 1\}$, tal que $w_{ci} = 1$ si el conductor c trabaja en el instante de tiempo i. Observar que w_{ci} podría estar activada aún cuando c no esté trabajando en i, sin embargo esto no será un inconveniente dado que en la realidad c estaría trabajando menos tiempo de lo dicho por el modelo. Además, se incorpora una variable $z_{ch} \in \{0, 1\}$ para cada $c \in C$ y $h \in \{0, \ldots, H - 1\}$, tal que $z_{ch} = 1$ si y solo si el conductor c está de franco en el día h. Estas variables deben estar sujetas a las siguientes restricciones.

- Si un conductor c al comienzo de su ruta realiza una tarea t que comienza en un instante i, entonces c trabaja desde el instante $i - \mathrm{tVT}(l_c, \mathrm{ori}(t))$ hasta el instante $i + \mathrm{dur}(t) - 1$:

$$\sum_{i' = i - \mathrm{tVT}(l_c, \mathrm{ori}(t))}^{i + \mathrm{dur}(t) - 1} w_{ci'} \geq (\mathrm{tVT}(l_c, \mathrm{ori}(t)) + \mathrm{dur}(t)).(x_{c(n_{l_c}, n_t)} + y_{ti} - 1) \\ \forall t \in T, \; c \in C, \; i \in \mathcal{I}_t$$

  Observar que estas restricciones programan el viaje en taxi inmediatamente antes del comienzo de t.

- Si un conductor c realiza una tarea t_2 que comienza en un instante i luego de una tarea t_1, entonces c trabaja desde el instante $i - \mathrm{tVT}(\mathrm{des}(t_1), \mathrm{ori}(t_2))$

hasta el instante $i + \text{dur}(t_2) - 1$:

$$\sum_{i'=i-\text{tVT}(\text{des}(t_1),\text{ori}(t_2))}^{i+\text{dur}(t_2)-1} w_{ci'} \geq (\text{tVT}(\text{des}(t_1), \text{ori}(t_2)) + \text{dur}(t_2)).(x_{c(n_{t_1},n_{t_2})} + y_{t_2 i} - 1) \qquad \forall t_1, t_2 \in T,\ t_1 \neq t_2,\ c \in C,\ i \in \mathcal{I}_{t_2}$$

Al igual que antes, el viaje en taxi es programado inmediatamente antes del comienzo de t_2.

- Todo conductor c trabaja a lo sumo 12 horas en todo intervalo de 24 horas:

$$\sum_{i'=i}^{i+I-1} w_{ci'} \leq \frac{I}{2} \qquad \forall c \in C,\ i \in \{0, \ldots, I.(H-1)\}$$

- Todo conductor c tiene al menos un día de franco en todo intervalo de 7 días:

$$\sum_{h' \in \{h,\ldots,h+6\}} z_{ch'} \geq 1 \qquad \forall c \in C,\ h \in \{0, \ldots, H-7\}$$

- Si un conductor c está de franco en el día h, entonces no trabaja durante ese día:

$$\sum_{i=I.h}^{I.h+I-1} w_{ci} \leq I.(1 - z_{ch}) \qquad \forall c \in C,\ h \in \{0, \ldots, H-1\}$$

2.2.3. Función objetivo

El único objetivo que se busca minimizar durante esta etapa es el costo por distancia recorrida en taxi. Nuevamente, para su modelado se considera una función costoTaxi : $E \to \mathbb{N}_0$ que asigna un costo a cada uno de los arcos de G_2 de acuerdo a este objetivo. La misma, se define de la siguiente forma:

$$\text{costoTaxi}(e) = \begin{cases} \text{costoViajeTaxi}(l, \text{ori}(t)) & \text{si } e = (n_l, n_t), \\ \text{costoViajeTaxi}(\text{des}(t_1), \text{ori}(t_2)) & \text{si } e = (n_{t_1}, n_{t_2}), \\ 0 & \text{caso contrario.} \end{cases}$$

Es decir, el costo de (n_l, n_t) es el de viajar en taxi desde la localidad inicial l a la localidad de origen de la tarea t. El costo de (n_{t_1}, n_{t_2}) es el de viajar en taxi desde la localidad de destino de la tarea t_1 a la de origen de la tarea t_2. Y finalmente, el costo es 0 para los demás arcos.

Luego, este objetivo se puede expresar como la suma de los pesos de los arcos recorridos por los conductores:

$$\text{mín} \sum_{c \in C} \sum_{e \in E_c} \text{costoTaxi}(e).x_{ce}$$

2.2.4. Formulación E2

A continuación se describe la formulación de PLE para el problema abordado en esta segunda etapa, cuyos elementos han sido presentados a lo largo de la sección.

$$\text{(E2)} \quad \text{mín} \sum_{c \in C} \sum_{e \in E_c} \text{costoTaxi}(e).x_{ce} \qquad (f_{\text{TAXI}})$$

Sujeto a:

$$\sum_{e \in \Gamma^+(n_{l_c})} x_{ce} = 1 \qquad \forall c \in C \quad (2.17)$$

$$\sum_{e \in \Gamma^-(n_t) \cap E_c} x_{ce} = \sum_{e \in \Gamma^+(n_t) \cap E_c} x_{ce} \qquad \forall c \in C,\ t \in T \quad (2.18)$$

$$1 \leq \sum_{c \in C} \sum_{e \in \Gamma^-(n_t) \cap E_c} x_{ce} \leq 2 \qquad \forall t \in T \quad (2.19)$$

$$\sum_{i \in \mathcal{I}_t} y_{ti} = 1 \qquad \forall t \in T \quad (2.20)$$

$$\sum_{c \in C} x_{ce} \leq |C|.X_e \qquad \forall e \in E \quad (2.21)$$

$$\delta(t) \geq \text{tVT}(l, \text{ori}(t)).X_{(n_l, n_t)} \qquad \forall l \in L_C,\ t \in T \quad (2.22)$$

$$\delta(t_2) \geq \delta(t_1) + \text{dur}(t_1) + \text{tVT}(\text{des}(t_1), \text{ori}(t_2)).X_{(n_{t_1}, n_{t_2})} \qquad \forall (t_1, t_2) \in T^{\Rightarrow} \quad (2.23)$$

$$\delta(t_2) \geq \delta(t_1) + \text{dur}(t_1) + \text{tVT}(\text{des}(t_1), \text{ori}(t_2)) - M.(1 - X_{(n_{t_1}, n_{t_2})}) \qquad \forall (t_1, t_2) \in T^{\neq} \quad (2.24)$$

$$\sum_{i' = i - \text{tVT}(l_c, \text{ori}(t))}^{i + \text{dur}(t) - 1} w_{ci'} \geq (\text{tVT}(l_c, \text{ori}(t)) + \text{dur}(t)).(x_{c(n_{l_c}, n_t)} + y_{ti} - 1) \qquad \forall t \in T,\ c \in C,\ i \in \mathcal{I}_t \quad (2.25)$$

$$\sum_{i'=i-\mathrm{tVT}(\mathrm{des}(t_1),\mathrm{ori}(t_2))}^{i+\mathrm{dur}(t_2)-1} w_{ci'} \geq (\mathrm{tVT}(\mathrm{des}(t_1),\mathrm{ori}(t_2)) + \mathrm{dur}(t_2)).(x_{c(n_{t_1},n_{t_2})} + y_{t_2 i} - 1) \quad \forall t_1, t_2 \in T,\ t_1 \neq t_2,\ c \in C,\ i \in \mathcal{I}_{t_2} \quad (2.26)$$

$$\sum_{i'=i}^{i+I-1} w_{ci'} \leq \frac{I}{2} \quad \forall c \in C,\ i \in \{0, \ldots, I.(H-1)\} \quad (2.27)$$

$$\sum_{h' \in \{h,\ldots,h+6\}} z_{ch'} \geq 1 \quad \forall c \in C,\ h \in \{0, \ldots, H-7\} \quad (2.28)$$

$$\sum_{i=I.h}^{I.h+I-1} w_{ci} \leq I.(1 - z_{ch}) \quad \forall c \in C,\ h \in \{0, \ldots, H-1\} \quad (2.29)$$

$$x_{ce} \in \{0,1\} \quad \forall c \in C,\ e \in E_c \quad (2.30)$$

$$X_e \in \{0,1\} \quad \forall e \in E \quad (2.31)$$

$$y_{ti} \in \{0,1\} \quad \forall t \in T,\ i \in \mathcal{I}_t \quad (2.32)$$

$$w_{ci} \in \{0,1\} \quad \forall c \in C,\ i \in \{0, \ldots, I.H-1\} \quad (2.33)$$

$$z_{ch} \in \{0,1\} \quad \forall c \in C,\ h \in \{0, \ldots, H-1\} \quad (2.34)$$

2.2.5. Subdigrafo de secuenciación de tareas

Algunos de los algoritmos que se presentarán en los próximos capítulos trabajan sobre un subdigrafo del digrafo de secuenciación de tareas, cuyos arcos deben ser compatibles con una programación horaria prefijada.

Formalmente, dada una programación horaria δ, se define el subdigrafo $G_2^\delta = (N, E^\delta)$ de $G_2 = (N, E)$, donde el subconjunto de arcos $E^\delta \subseteq E$ contiene únicamente las transiciones que pueden realizarse cuando las tareas se programan de acuerdo a δ. Más precisamente, E^δ tiene los siguientes arcos:

- (n_l, n_t) para cada $l \in L_C$ y $t \in T$ tal que el horario de la tarea t le permite a un conductor tomar un taxi desde su localidad inicial hasta la de origen de t, es decir, $\mathrm{tiempoViajeTaxi}(l, \mathrm{ori}(t)) \leq \delta(t)$.
- (n_{t_1}, n_{t_2}) para cada $t_1, t_2 \in T$ con $t_1 \neq t_2$ tal que la tarea t_2 comienza luego de que t_1 finalice y además le permite a un conductor tomar un taxi desde la localidad destino de t_1 a la de origen de t_2, es decir, $\delta(t_1) + \mathrm{dur}(t_1) + \mathrm{tiempoViajeTaxi}(\mathrm{des}(t_1), \mathrm{ori}(t_2)) \leq \delta(t_2)$.
- (n_l, n_s) para cada $l \in L_C$ y (n_t, n_s) para cada $t \in T$.

A continuación, se provee un ejemplo de esta construcción.

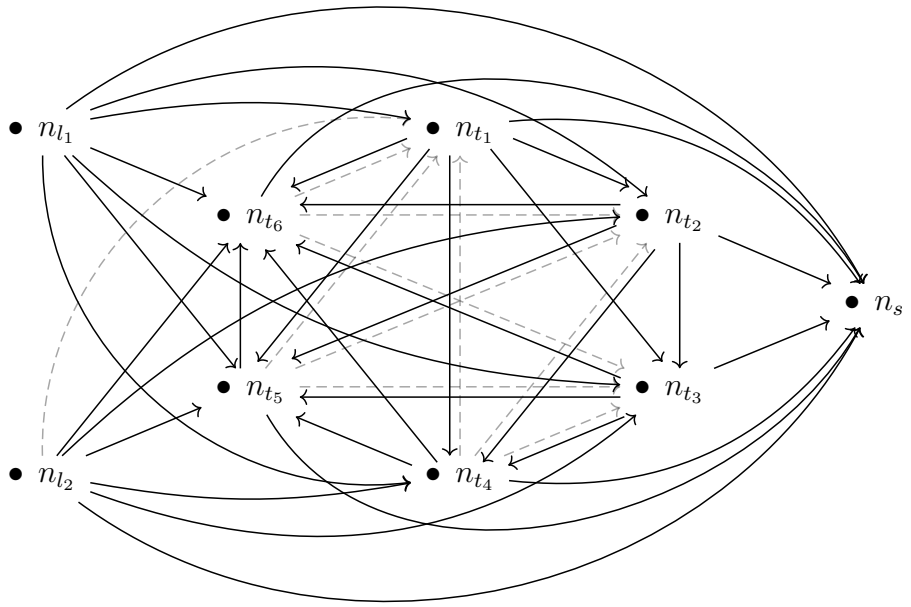


Figura 2.2.4.: Digrafo de secuenciación de tareas para el Ejemplo 1.2.1 dada la programación horaria δ del Ejemplo 2.1.7.

Ejemplo 2.2.4. En la Figura 2.2.4, se muestra el digrafo de secuenciación de tareas que resulta para el Ejemplo 1.2.1 dada la programación horaria definida en el Ejemplo 2.1.7. Los arcos en línea discontinua y de color más claro son los arcos de $E(G_2)$ que ya no pertenecen a $E(G_2^\delta)$.

Por su construcción, es fácil notar que G_2^δ es un *grafo acíclico dirigido* DAG[1]. Por lo tanto, todo (n_{l_c}, n_s)-camino dirigido se corresponde con una secuencia de tareas que c puede realizar de manera consecutiva en los horarios programados por δ, aunque aún es necesario comprobar que se cumplan las regulaciones laborales.

[1]Del inglés, Directed Acyclic Graph

3. Resolución heurística

En este capítulo, se describen los algoritmos de optimización desarrollados para resolver los dos subproblemas provenientes de la descomposición secuencial. Son de carácter heurísticos y/o metaheurístico, dado que el abordaje secuencial es naturalmente no-exacto, es decir, las soluciones óptimas de cada subproblema no necesariamente conducen a soluciones globalmente óptimas. Por ejemplo, las decisiones tomadas en la primera etapa pueden restringir demasiado el espacio de búsqueda, haciendo que las soluciones óptimas de la segunda sean de muy mala calidad e incluso el segundo subproblema podría ser infactible para la instancia proveniente de la solución óptima del primero.

Para la resolución del primer subproblema, el cual es una variante de un PDPTW, se aplica una adaptación de la metaheurística híbrida propuesta por [Ropke and Pisinger, 2006], que combina *Large Neighborhood Search* (LNS) con un criterio de aceptación proveniente de *Simulated Annealing* (SA). El método de reparación consiste en una heurística de inserción paralela de k-arrepentimiento randomizada con ruido en la función objetivo. Mientras que para el método de destrucción, se describen tres heurísticas diferentes.

Para la resolución del segundo subproblema, se propone un *Greedy Randomized Adaptive Search Procedure* (GRASP), donde en la fase de construcción se aplica una heurística de inserción paralela randomizada por medio de una lista de candidatos restringida, y en la fase de búsqueda local se consideran múltiples tipos de movimientos, algunos estándares y otros específicos al caso de estudio. Progresivamente, se incorporan dos nuevas funcionalidades al GRASP a fin de mejorar su rendimiento. La primera de ellas facilita el arranque, permitiendo violar algunas regulaciones laborales durante la construcción de la solución inicial, y reparándolas luego con una heurística. Mientras que la otra, realiza una perturbación de la

programación horaria de las tareas para escapar de óptimos locales, dando lugar así a una metaheurística híbrida GRASP×ILS.

3.1. Etapa I – Resolución mediante LNS

Al considerar una descomposición secuencial para el caso de estudio, el primero de los subproblemas involucrados es un PDPTW (ver Sección 2.1). Este problema es una de las variaciones más estudiadas de VRPs, existiendo en la literatura gran número de trabajos enfocados en su estudio y resolución [Toth and Vigo, 2014]. En esta tesis, se optó por elegir uno de estos enfoques y adaptarlo al caso de estudio particular.

La elegida fue la metaheurística híbrida de [Ropke and Pisinger, 2006], que mostró resultados alentadores para un problema bastante similar. Ella combina *Large Neighborhood Search* (LNS) [Shaw, 1998, Pisinger and Ropke, 2019], con un criterio de aceptación proveniente de *Simulated Annealing* [Kirkpatrick et al., 1983, Delahaye et al., 2019]. Los esquemas clásicos para estas metaheurísticas se detallan en los Apéndices 13 y 4.3, respectivamente. Sin embargo, adaptar esta implementación al caso de estudio no es trivial y en esta sección se mencionan los principales desafíos afrontados.

Dado que existen dos objetivos relacionados a los recorridos de los camiones, se propone como primera función objetivo la suma ponderada de ambos. Siendo *sol* una solución, la fórmula aplicada es

$$f(sol) = \lambda.f_{\text{ENTR}}(sol) + (1 - \lambda).f_{\text{VIAJ}}(sol), \tag{3.1}$$

donde $\lambda \in [0, 1]$ determina la importancia de cada uno de los objetivos en el valor final, y las funciones f_{ENTR} y f_{VIAJ} retornan la penalidad por la demora en las entregas programadas y el costo por la distancia recorrida por los camiones para dicha solución, respectivamente. Dado que no existe un valor ideal para el parámetro λ, sino que depende de qué tanto el tomador de decisiones prefiera priorizar uno de los objetivos sobre el otro, se espera que la heurística encuentre buenas soluciones independientemente del valor elegido.

El pseudocódigo de la metaheurística híbrida se muestra en el Algoritmo 1. En el mismo intervienen tres parámetros: $t_{frio} \in (0,1)$ es la tasa de enfriamiento que regula la velocidad con la que se enfría el sistema, $t_0 \in (0,1)$ controla la temperatura inicial del sistema y $nIters \in \mathbb{N}$ es el número total de iteraciones a realizar. Los valores más adecuados para los mismos serán determinados más adelante por medio de experimentación computacional (al igual que para el resto de los parámetros que intervendrán en la reparación y la destrucción de las soluciones). La solución inicial requerida como entrada se construye con el mismo método de reparación que se presentará en la próxima subsección.

Algoritmo 1 `LNS`

Entrada: sol : solución.
Salida: sol^* : solución.
Parámetros: $t_{frio} \in (0,1)$, $t_0 \in (0,1)$, e $nIters \in \mathbb{N}$.

1: $f(sol^*) \leftarrow +\infty$
2: $T \leftarrow \frac{t_0}{\ln(0,5)}.f(sol)$
3: **for** $it \leftarrow 0;\ it < nIters;\ it \leftarrow it + 1$ **do**
4: $\quad T \leftarrow T.t_{frio}$
5: $\quad P', sol'' \leftarrow$ `destruir`(sol)
6: $\quad$ **try**
7: $\quad\quad sol' \leftarrow$ `reparar`(P', sol'')
8: $\quad$ **catch** *falloReparar*
9: $\quad\quad$ **continue**
10: $\quad$ **if** `rand`$([0,1]) \leq e^{\frac{-(f(sol')-f(sol))}{T}}$ **then**
11: $\quad\quad sol \leftarrow sol'$
12: $\quad$ **if** $f(sol') < f(sol^*)$ **then**
13: $\quad\quad sol^* \leftarrow sol'$
14: **return** sol^*

En la línea 1, se inicializa el valor objetivo de la mejor solución encontrada sol^* como infinito. En la línea 2, se determina la temperatura inicial del sistema, de forma tal que haya 50 % de probabilidad de aceptar una nueva solución cuyo valor objetivo sea t_0 veces peor que el valor objetivo de la solución inicial. En la línea 4, se ajusta la temperatura del sistema, multiplicándola por t_{frio}. En la línea 5, se aplica una heurística de destrucción a la solución actual sol, retornando una solución parcial sol'' y el subconjunto de pedidos P' que no fueron removidos de sol. En las líneas 6–7, se intenta aplicar una heurística de reparación para reinsertar en sol'' todos los pedidos de $P \setminus P'$. La reparación puede devolver una nueva solución sol', o bien puede lanzar una excepción *falloReparar*, que indica que no fue posible repararla. En caso de capturar una excepción, se prosigue con

la siguiente iteración del bucle de la línea 3. De lo contrario, en las líneas 10–11, se actualiza sol por sol' según el criterio de aceptación proveniente de SA. La función `rand` toma un conjunto y retorna un elemento aleatorio del mismo. En las líneas 12–13, se actualiza sol^* por sol' en caso de que haya una mejora en el valor objetivo. En la línea 14, se retorna sol^* tras alcanzar el límite de iteraciones.

A continuación, se presentan diferentes heurísticas para reparar y destruir las soluciones. Cabe aclarar que la implementación original de [Ropke and Pisinger, 2006] es un poco más sofisticada que la aquí presentada. En particular, en cada iteración se determina de manera adaptativa cuál heurística aplicar, teniendo en cuenta la efectividad que cada una de ellas demostró en las últimas iteraciones. En nuestro caso, ese comportamiento adaptativo no fue incorporado, debido a que la calidad de las soluciones encontradas utilizando siempre una misma heurística (fijada de antemano) ya era aceptable. Más adelante, se presentarán experimentos computacionales para sustentar esta afirmación.

3.1.1. Método de construcción y reparación

Tanto para construir soluciones iniciales desde cero, como para completar soluciones incompletas, es decir, repararlas luego de ser destruidas, se aplica una heurística de inserción paralela de k-arrepentimiento, randomizada con ruido en la función objetivo, que es una adaptación de la propuesta por [Ropke and Pisinger, 2006]. En particular, las heurísticas de arrepentimiento se han popularizado en la construcción de rutas para problemas del área, e.g. [Diana and Dessouky, 2004, Hemmelmayr et al., 2012]. El pseudocódigo completo de la misma se omite ya que consideramos que no reviste mayores dificultades, aunque las ideas más importantes se describen en el Apéndice 4.1.

A continuación, se presenta un fragmento del mismo encargado de computar el (mejor) costo de inserción de un pedido en un ruta. En nuestro caso, y a diferencia de [Ropke and Pisinger, 2006], este costo no sólo depende de la posición en donde se inserta (¿entre qué pedidos?), sino también del horario de su entrega. Esto se debe a que la función objetivo (3.1) precisa de este dato para computar la penalidad correspondiente por las entregas tardías. Esto quiere decir que, insertar un pedido

p en una ruta entre dos pedidos p_1 y p_2, puede tener diferente costo dependiendo del día en que se entregue p.

El pseudocódigo de la función que calcula el costo de inserción se muestra en el Algoritmo 2. La entrada del mismo es una solución *sol*, un pedido p a insertar, un camión v a realizar el nuevo pedido y el valor de λ. Los parámetros $ruido_{\text{VIAJ}} \in \mathbb{R}_+$ y $ruido_{\text{ENTR}} \in \mathbb{R}_+$ escalan el ruido aplicado a cada objetivo. El ruido máximo M_{VIAJ} aplicado a f_{VIAJ} es el máximo costo de viaje en camión entre todos los pares de localidades, i.e. $M_{\text{VIAJ}} = \text{máx}_{l,l' \in L}\, \text{costoViajeCamion}(l, l')$, y el ruido máximo M_{ENTR} aplicado a f_{ENTR} es la máxima penalidad por un día de demora entre todos los pedidos, i.e. $M_{\text{ENTR}} = \text{máx}_{p \in P}\, w_p$.

Algoritmo 2 `costoInserción`

Entrada: *sol* : solución, $p \in P$, $v \in V$, y $\lambda \in [0, 1]$.
Salida: $ci^* \in \mathbb{R}$.
Parámetros: $ruido_{\text{ENTR}} \in \mathbb{R}_+$ y $ruido_{\text{VIAJ}} \in \mathbb{R}_+$.

1: $r_v,\ i^{\text{R}},\ i^{\text{E}} \leftarrow sol$
2: $ci^* \leftarrow +\infty$
3: **for** $j \leftarrow 1;\ j < \texttt{lon}(r_v);\ j \leftarrow j + 1$ **do**
4: $\quad \Delta f_{\text{VIAJ}} \leftarrow \text{costoCamion}(r_v[j-1], n_p) + \text{costoCamion}(n_p, r_v[j]) - \text{costoCamion}(r_v[j-1], r_v[j])$
5: $\quad$ **if** $r_v[j-1] = n_{l_v}$ **then**
6: $\quad\quad ini \leftarrow \text{tiempoViajeCamion}(l_v, l_p^{\text{R}})$
7: $\quad$ **else if** $r_v[j-1] = n_{p'}$ para algún $p' \in P$ **then**
8: $\quad\quad ini \leftarrow i_{p'}^{\text{E}} + \text{tiempoViajeCamion}(l_{p'}^{\text{E}}, l_p^{\text{R}})$
9: $\quad$ **if** $r_v[j] = n_s$ **then**
10: $\quad\quad fin \leftarrow I.H$
11: $\quad$ **else if** $r_v[j] = n_{p''}$ para algún $p'' \in P$ **then**
12: $\quad\quad fin \leftarrow i_{p''}^{\text{R}} - \text{tiempoViajeCamion}(l_p^{\text{E}}, l_{p''}^{\text{R}})$
13: $\quad$ **for** $i_p^{\text{R}} \leftarrow ini;\ i_p^{\text{R}} + s_p^{\text{R}} + \text{tiempoViajeCamion}(l_p^{\text{R}}, l_p^{\text{E}}) + s_p^{\text{E}} \le fin;\ i_p^{\text{R}} \leftarrow i_p^{\text{R}} + 1$ **do**
14: $\quad\quad$ **if** $i_p^{\text{R}} \notin \mathcal{I}_p^{\text{R}}$ **then continue**
15: $\quad\quad$ **for** $i_p^{\text{E}} \leftarrow i_p^{\text{R}} + s_p^{\text{R}} + \text{tiempoViajeCamion}(l_p^{\text{R}}, l_p^{\text{E}});\ i_p^{\text{E}} + s_p^{\text{E}} \le fin;\ i_p^{\text{E}} \leftarrow i_p^{\text{E}} + 1$ **do**
16: $\quad\quad\quad$ **if** $i_p^{\text{E}} \notin \mathcal{I}_p^{\text{E}}$ **then continue**
17: $\quad\quad\quad \Delta f_{\text{ENTR}} \leftarrow (\text{dia}(i_p^{\text{E}}) - dia_p^{\text{E}}).w_p$
18: $\quad\quad\quad ci \leftarrow \lambda.(\Delta f_{\text{ENTR}} + ruido_{\text{ENTR}}.\texttt{rand}([-M_{\text{ENTR}}, M_{\text{ENTR}}])) + (1-\lambda).(\Delta f_{\text{VIAJ}} + ruido_{\text{VIAJ}}.\texttt{rand}([-M_{\text{VIAJ}}, M_{\text{VIAJ}}]))$
19: $\quad\quad\quad ocu \leftarrow i_p^{\text{E}} + s_p^{\text{E}} - i_p^{\text{R}}$
20: $\quad\quad\quad$ **if** $ci < ci^*$ **or** $(ci = ci^*$ **and** $ocu < ocu^*)$ **then**
21: $\quad\quad\quad\quad ci^*, ocu^* \leftarrow ci, ocu$
22: **return** ci^*

En la línea 1, se extraen de la solución los siguientes elementos: un arreglo r_v

con los nodos del (n_{l_v}, n_s)-camino dirigido de G_1 que representan la ruta de v (si v aún no tiene pedidos asignados, se considera que existe en G_1 un arco ficticio (n_{l_v}, n_s) de costo 0) y funciones i^{R} e i^{R} que devuelven los instantes de tiempo en que comienzan la recolección y la entrega de cada pedido cumplido por v. En el bucle de la línea 3, se recorre cada índice j de r y se analiza la opción de insertar el nodo n_p entre los nodos r_{j-1} y r_j. Observar que j no toma los valores 0 ni $\mathrm{lon}(r)$, pues n_p no puede insertarse antes que n_{l_v} ni después que n_s. En las líneas 5–8, se determina el primer instante de tiempo en el que puede comenzar la recolección de p, el cual depende de si p es el primer pedido de la ruta o si debe realizarse tras entregar otro pedido p'. De manera análoga, en las líneas 9–12, se determina el último instante de tiempo en que puede finalizar la entrega de p, el cual depende de si p es el último pedido de la ruta o si debe realizarse antes de recolectar otro pedido p''. En los bucles de las líneas 13 y 15, se analizan cada uno de los instantes de tiempo en los que puede comenzar la recolección y la entrega de p, considerando los límites calculados anteriormente. Además, en las líneas 14 y 16, se verifican que las respectivas ventanas de tiempo estén abiertas. En las líneas 20–21, se actualizan las variables en caso de haber encontrado una mejor posición, y en caso de empate, se elige la posición en la cual el camión está ocupado el menor tiempo (calculado en la línea 19).

3.1.2. Métodos de destrucción

En este trabajo, se consideran tres heurísticas de destrucción diferentes, las cuales también se utilizan en [Ropke and Pisinger, 2006]. Todas ellas están parametrizadas por $\alpha \in (0, 1]$, que representa la máxima proporción de pedidos que serán removidos de la solución. En términos generales, estas heurísticas toman como entrada una solución *sol*, seleccionan con algún criterio una cantidad aleatoria $n \leq \alpha.|P|$ de pedidos y los remueven de la solución. El criterio con el que se eligen los pedidos a remover es lo que varía en cada una de las diferentes heurísticas.

Eliminación de Shaw

Se enfoca en remover pedidos que guarden algún tipo de relación entre ellos y fue propuesta originalmente por P. Shaw en [Shaw, 1997, Shaw, 1998]. Para medir qué

tan parecidos son dos pedidos p y p', se define la *medida de relación* $r(p, p') \geq 0$. Un valor bajo de $r(p, p')$ indica que los pedidos están muy relacionados.

En nuestra adaptación, la medida de relación utilizada es la suma ponderada entre un término de distancia, que cuantifica la cercanía entre las localidades de recolección y de entrega de p con las de p', y de un término de tiempo, que cuantifica la cercanía entre los horarios programados para la recolección y la entrega de p con los de p'. Específicamente, se define

$$r(p, p') = \lambda_{shaw}.\frac{\mathrm{tVC}(l_p^{\mathrm{R}}, l_{p'}^{\mathrm{R}}) + \mathrm{tVC}(l_p^{\mathrm{E}}, l_{p'}^{\mathrm{E}})}{2.M} + (1 - \lambda_{shaw}).\frac{|i_p^{\mathrm{R}} - l_{p'}^{\mathrm{R}}| + |i_p^{\mathrm{E}} - l_{p'}^{\mathrm{E}}|}{2.H},$$

donde i^{R} e i^{E} son funciones dependientes de la solución que retornan el horario programado para la recolección y la entrega de cada pedido, $\lambda_{shaw} \in [0, 1]$ es un parámetro que pondera la importancia de cada uno de los términos en el valor final y M es la constante *big-M* usada en la formulación E1 (i.e. $M = I.H$). Notar además que ambos términos se encuentran normalizados.

El pseudocódigo de esta heurística se presenta en el Algoritmo 3. Además de los parámetros ya mencionados, también se encuentra el *parámetro de determinismo* $e_{shaw} \geq 1$ que randomiza la elección de los pedidos a remover. En particular, con valores de e_{shaw} cercanos a 1, la elección será más aleatoria y a medida que su valor se incremente, la elección estará cada vez más guiada por la medida de relación.

Algoritmo 3 `destruirShaw`

Entrada: sol : solución.
Salida: $P' \subset P$ y sol' : solución parcial.
Parámetros: $\alpha \in (0, 1]$, $\lambda_{shaw} \in [0, 1]$, y $e_{shaw} \in \mathbb{R}_{\geq 1}$.

1: $i^{\mathrm{R}}, i^{\mathrm{E}} \leftarrow sol$
2: $r \leftarrow \texttt{rand}(\{4, \ldots, \text{mín}(100, \alpha.|P|)\})$
3: $p_0 \leftarrow \texttt{rand}(P)$
4: $R \leftarrow \{p_0\}$
5: **while** $|R| < r$ **do**
6: $\quad p \leftarrow \texttt{rand}(R)$
7: $\quad arr \leftarrow \texttt{ordenar}(P \setminus R)$ tal que $p' \leq p''$ si $r(p, p') \leq r(p, p'')$
8: $\quad i \leftarrow \lfloor \texttt{rand}([0, 1))^{e_{shaw}}.|P \setminus R| \rfloor$
9: $\quad R \leftarrow R \cup \{arr[i]\}$
10: **for** $p \in R$ **do**
11: $\quad sol' \leftarrow \texttt{eliminar}(sol, p)$
12: **return** $P \setminus R,\ sol'$

En la línea 1, se extraen de la solución las funciones antes mencionadas: i^{R} e i^{E}. En la línea 2, se elige una cantidad r de pedidos a remover, que es un número aleatorio entre 4 y mín$(100, \alpha.|P|)$ (remover menos de 4 pedidos haría muy difícil que puedan ser reinsertados en posiciones diferentes a las originales y remover más de 100, haría que los tiempos de ejecución se incrementen demasiado). En las líneas 3-4, se elige un pedido inicial $p_0 \in P$ de manera aleatoria y se lo inserta en un conjunto vacío R, que contendrá los pedidos a remover. En las líneas 6-7, se elige un pedido aleatorio p de R y se ordenan los pedidos de $P \backslash R$ en orden ascendente según su medida de relación respecto a p, dando como resultado un arreglo ordenado arr. En las líneas 8-9, se elige un índice i de manera semialeatoria y se inserta en R el pedido $arr[i]$. Para la elección de i se utiliza la fórmula

$$\lfloor \texttt{rand}([0,1))^{e_{shaw}}.|P \setminus R| \rfloor,$$

resultando la elección más golosa cuando e_{shaw} se acerca a 1.

En la práctica, no es necesario realizar el costoso ordenamiento de arr, sino que basta con aplicar un algoritmo de selección de tiempo lineal [Cormen et al., 2009].

Eliminación de peores

Se enfoca en remover pedidos cuyas inserciones hayan generado grandes incrementos en el valor objetivo de la solución. El costo de remover un pedido p de una solución sol está dado por

$$\texttt{costoRemocion}(sol, p) = f(sol) - f(sol'),$$

donde sol' es la solución parcial sin el pedido p.

El pseudocódigo de la heurística se muestra en el Algoritmo 4 y se aplican las mismas ideas que en la eliminación de Shaw. En este caso, el parámetro de determinismo se llama e_{peor} y los pedidos se ordenan descendentemente de acuerdo a su costo de remoción (líneas 4-5).

Algoritmo 4 `destruirPeor`

Entrada: sol : solución.
Salida: $P' \subset P$ y sol' : solución parcial.
Parámetros: $\alpha \in (0, 1]$ y $e_{peor} \in \mathbb{R}_{\geq 1}$.

1: $r \leftarrow \texttt{rand}(\{4, \ldots, \text{mín}(100, \alpha.|P|)\})$
2: $R \leftarrow \emptyset$
3: **while** $|R| < r$ **do**
4: $\quad arr \leftarrow \texttt{ordenar}(P \setminus R)$ tal que
$\qquad\qquad p \leq p'$ si $\texttt{costoRemoción}(sol, p) \geq \texttt{costoRemoción}(sol, p')$
5: $\quad i \leftarrow \lfloor \texttt{rand}([0, 1))^{e_{peor}}.|P \setminus R| \rfloor$
6: $\quad R \leftarrow R \cup \{arr[i]\}$
7: $\quad sol' \leftarrow \texttt{eliminar}(sol, arr[i])$
8: **return** $P \setminus R,\ sol'$

Eliminación aleatoria

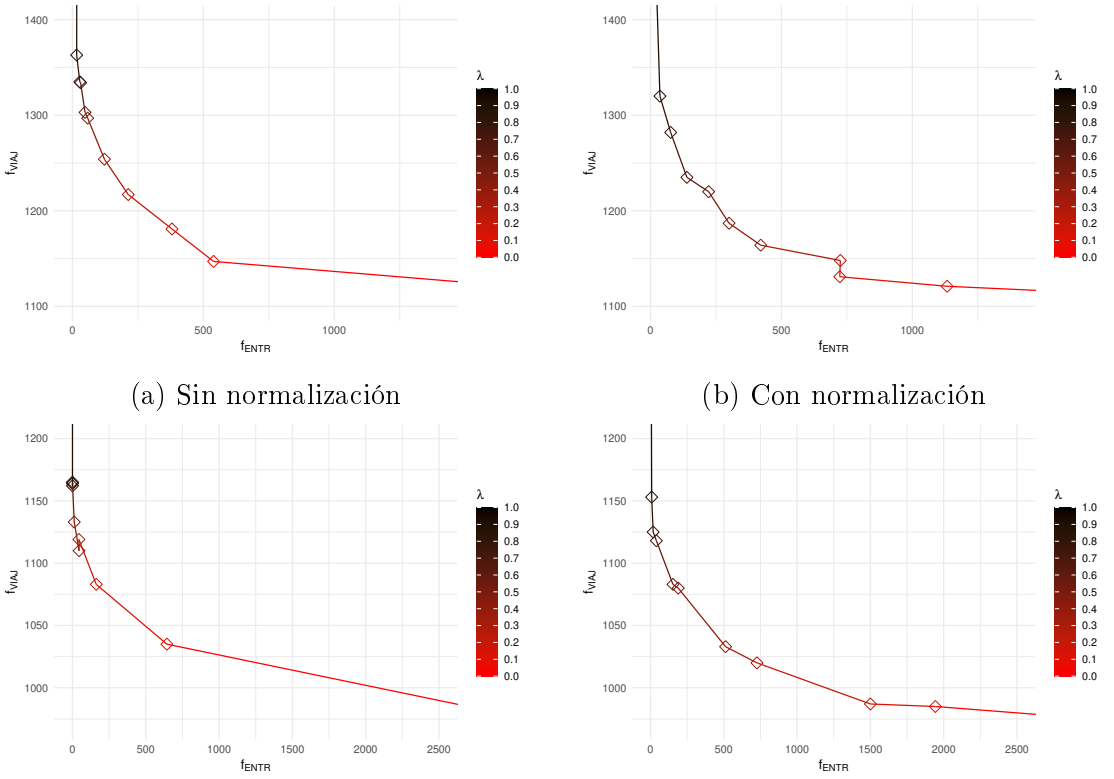
Los pedidos a remover se eligen de manera puramente aleatoria. A pesar de ser un caso particular de la eliminación de Shaw con $e_{shaw} = 1$ y de la eliminación de peores con $e_{peor} = 1$, se implementó por razones de eficiencia un algoritmo separado. Por su simplicidad, se omite el pseudocódigo del mismo.

3.1.3. Normalización

Considerando que las funciones objetivo tienen diferentes unidades, se decide normalizarlas usando los puntos ideal y nadir [Ehrgott, 2005]. Un resumen de estas técnicas se menciona en el Apéndice 5. En la implementación, estos puntos se estiman con la propia metaheurística, dado que no es posible computarlos de forma exacta para el tamaño de instancias usado en los experimentos computacionales.

Más específicamente, para cada instancia, se realiza un preprocesamiento donde se ejecuta la metaheurística con $\lambda = 0$ y $\lambda = 1$, y un tiempo límite de 1 minuto. Siendo sol_0^* y sol_1^* las soluciones generadas, respectivamente, la función objetivo normalizada para esa instancia queda de la siguiente forma, siendo λ un valor arbitrario en el intervalo $[0, 1]$:

$$\tilde{f}(sol) = \lambda.\frac{f_{\text{ENTR}}(sol) - f_{\text{ENTR}}(sol_1^*)}{f_{\text{ENTR}}(sol_0^*) - f_{\text{ENTR}}(sol_1^*)} + (1 - \lambda).\frac{f_{\text{VIAJ}}(sol) - f_{\text{VIAJ}}(sol_0^*)}{f_{\text{VIAJ}}(sol_1^*) - f_{\text{VIAJ}}(sol_0^*)}. \quad (3.2)$$



(a) Sin normalización

(b) Con normalización

(c) Sin normalización

(d) Con normalización

Figura 3.1.1.: Aproximación de la frontera de Pareto.

Los resultados obtenidos para dos instancias particulares se muestran en la Figura 3.1.1. Las gráficas superiores corresponden a los resultados de la primera instancia y las inferiores a la segunda, además las gráficas de la izquierda no usan normalización y las de la derecha sí. Cada gráfica muestra la aproximación de la frontera de Pareto basada en las 11 soluciones que fueron devueltas por la metaheurística al variar λ entre 0 y 1 con un paso de 0,1 en la función objetivo (3.2) (el valor de cada solución se marca con un rombo, cuyo color depende del valor de λ).

Se puede observar que los objetivos están claramente en conflicto, es decir, mientras más se intenta minimizar uno de ellos, más empeora el otro. También es evidente que las funciones objetivo se mueven en rangos de valores diferentes. En las gráficas sin normalización, los puntos tienden a estar más cerca del eje vertical, lo que indica que f_{ENTR} tiende a priorizarse por sobre f_{VIAJ}. Por el contrario, en las gráficas con normalización, los puntos quedan distribuidos de manera un poco más uniforme sobre la aproximación de la frontera de Pareto. En particular, para valores de λ

cercanos a 0, la normalización permite encontrar soluciones con menor costo de distancia que las encontradas sin normalizar.

Lo que resta del capítulo se enfoca en la resolución heurística de la siguiente etapa del problema.

3.2. Etapa II – Resolución mediante GRASP

El segundo de los subproblemas consiste en planificar las tripulaciones de las tareas (ver Sección 2.2). Para su resolución, se propone una metaheurística Greedy Randomized Adaptive Search Procedure (GRASP) [Feo and Resende, 1989, Feo and Resende, 1995, Resende and Ribeiro, 2016]. En la literatura, es posible encontrar un gran número de trabajos que estudian la aplicación de esta metaheurística a diferentes problemas de optimización y que dan cuenta de su efectividad [Resende and Ribeiro, 2019]. Particularmente para VRPs, se pueden mencionar los siguientes trabajos de interés [Kontoravdis and Bard, 1995, Carreto and Baker, 2002, Ho and Szeto, 2016, Al Chami et al., 2018].

Una solución *sol* para este subproblema puede representarse de manera simplificada como un par $(\delta, \{r_c : c \in C\})$, donde δ es una programación horaria para las tareas de T y, para cada conductor c, r_c es un (n_{l_c}, n_s)-camino dirigido de G_2. Por supuesto, la solución debe verificar que cada nodo de tarea sea visitado por al menos uno y a lo sumo dos caminos y que se cumplan las restricciones de descanso.

La función objetivo a minimizar en esta etapa, notada f_{TAXI}, retorna el costo por la distancia total recorrida en taxi en una solución dada. Es decir, dada una solución $sol = (\delta, \{r_c : c \in C\})$,

$$\begin{aligned} f_{\text{TAXI}}(r_c) &= \sum_{i=0}^{\texttt{lon}(r_c)-1} \text{costoTaxi}(r_c[i], r_c[i+1]), \\ f_{\text{TAXI}}(sol) &= \sum_{c \in C} f_{\text{TAXI}}(r_c). \end{aligned} \tag{3.3}$$

El pseudocódigo general de GRASP se encuentra detallado en el Apéndice 4.3. A continuación, se describen las heurísticas propuestas para las fases de construcción y de mejora.

3.2.1. Fase de construcción

La construcción de la solución inicial se realiza con una heurística de inserción paralela de mejores, randomizada por medio de un parámetro de calidad $\alpha \in [0, 1]$ (ver Apéndice 4.1). El pseudocódigo de la heurística constructiva se presenta en el Algoritmo 5.

Algoritmo 5 `construir`

Entrada: T: conjunto de tareas y $\delta : T \to \mathcal{I}$ programación horaria.
Salida: $\{r_c : c \in C\}$: conjunto de rutas de conductor.
Parámetros: $\alpha \in [0, 1]$.

1: $T^{ord} \leftarrow$ `ordenar`(T) tal que $t \leq t'$ si $\delta(t) \leq \delta(t')$
2: $r_c \leftarrow \{n_{l_c}\}$, para cada $c \in C$
3: **for all** $t \in T^{ord}$ **do**
4: $\quad LC \leftarrow \{c \in C : r_c \cup \{n_t\}$ es un camino dirigido de G_2^δ
$\qquad\qquad \wedge\ c$ cumple restricciones de descanso$\}$
5: $\quad$ **if** $LC = \emptyset$ **then** fail
6: $\quad ci_c \leftarrow f_{\text{TAXI}}(r_c \cup \{n_t\}) - f_{\text{TAXI}}(r_c)$, para cada $c \in LC$
7: $\quad ci_{min}, ci_{max} \leftarrow \text{mín}_{c \in LC}\, ci_c,\ \text{máx}_{c \in LC}\, ci_c$
8: $\quad LCR \leftarrow \{c \in LC : ci_c \leq ci_{min} + \alpha.(ci_{max} - ci_{min})\}$
9: $\quad c^* \leftarrow$ `rand`(LCR)
10: $\quad r_{c^*} \leftarrow r_{c^*} \cup \{n_t\}$
11: $r_c \leftarrow r_c \cup \{n_s\}$, para cada $c \in C$
12: **return** $\{r_c : c \in C\}$

En la línea 1, se ordena el conjunto de tareas en orden ascendente según la hora a la que están programadas. En la línea 2, para cada conductor c, se inicia el camino dirigido r_c con un único nodo correspondiente a su localidad inicial. En las líneas 4-5, se computa la lista de candidatos LC con todos los conductores que pueden realizar la tarea t sin incumplir con las regulaciones laborales, fallando si LC es vacía. En la línea 6, para cada candidato c, se computa el costo de inserción de t al final de su ruta. En las líneas 7–8, se computa la lista de candidatos restringida LCR. En las líneas 9–10, se elige un conductor aleatorio c^* de LCR y se extiende a r_{c*} con el nodo n_t. En la línea 11, se extiende cada uno de los caminos con el nodo sumidero.

Como se ha mencionado, la heurística constructiva falla cuando LC es vacía, ya que ningún conductor podría realizar t sin comprometer la factibilidad de la solución. En ese caso, se sigue con la siguiente iteración del GRASP; en la siguiente sección se propondrá una heurística de reparación.

3.2.2. Fase de mejora

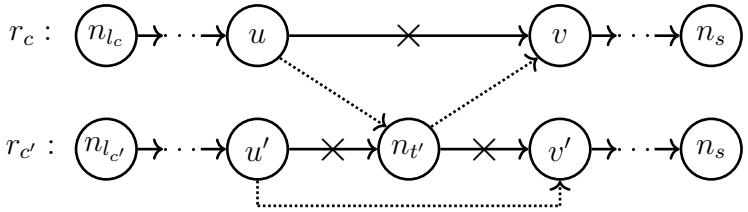
Para mejorar la solución construida en la fase previa se propone una búsqueda local de tipo *Variable Neighborhood Descent* (VND) [Mladenović and Hansen, 1997, Hansen et al., 2019]. Esta metaheurística ha sido aplicada a numerosos problemas de optimización, para una revisión más detallada consultar en [Hansen and Mladenović, 2001, Hansen et al., 2010]. El esquema general de VND se describe en detalle en el Apéndice 4.2.

Los primeros tres movimiento considerados son bastante comunes en la literatura (e.g. ver [Shen and Kwan, 2001, Ma et al., 2016]) y consisten en seleccionar dos conductor c y c' e intercambiar ciertos nodos entre sus rutas r_c y $r_{c'}$, respectivamente. Los dos movimientos restantes son específicos del caso de estudio y permiten modificar el número de tripulantes de una tarea. Estos últimos, seleccionan un conductor c y copian/borran nodos en su ruta r_c. Formalmente, la entrada de estos movimientos es una solución factible $sol = (\delta, \{r_c : c \in C\})$ y el resultado es una nueva solución $sol' = (\delta, \{r'_c : c \in C\})$. Ninguno de los movimientos presentados modifica la programación horaria δ; más adelante se presenta un nuevo movimiento para tal propósito.

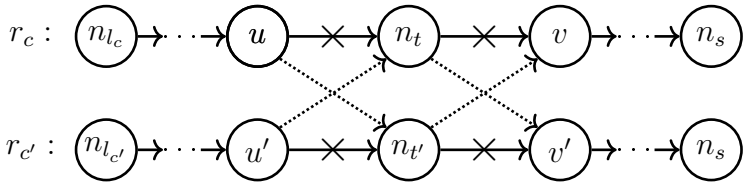
A continuación, se brinda una descripción detallada de cada uno de ellos. Además, en la Figura 3.2.1, se ilustra el resultado de su aplicación, donde los arcos que tienen una cruz en el medio son aquellos que deben removerse y los arcos en línea discontinua son los que deben insertarse.

Mov. 1: reubicar una tarea. Para $c, c' \in C$ y $t' \in T$, quitar el nodo $n_{t'}$ de $r_{c'}$ y reubicarlo en r_c (ver Figura 3.2.1a).

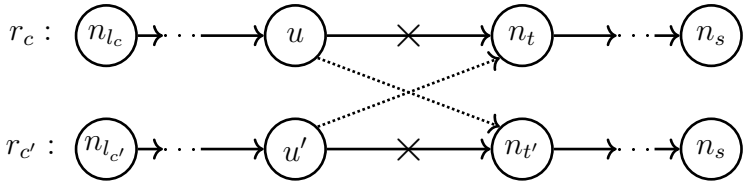
Mov. 2: intercambiar dos tareas. Para $c, c' \in C$ y $t, t' \in T$, intercambiar el nodo n_t de r_c con el nodo $n_{t'}$ de $r_{c'}$ (ver Figura 3.2.1b).



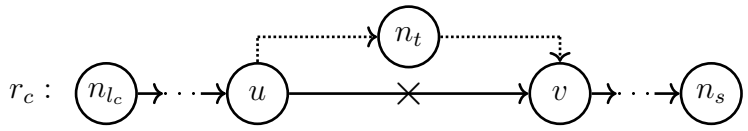
(a) Reubicar una tarea.



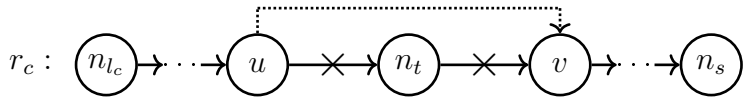
(b) Intercambiar dos tareas.



(c) Intercambiar dos arcos.



(d) Insertar una tarea.



(e) Remover una tarea.

Figura 3.2.1.: Movimientos.

Mov. 3: intercambiar dos arcos. Para $c, c' \in C$ y $t, t' \in T$, intercambiar todos los nodos de r_c a partir de n_t por todos los nodos de $r_{c'}$ a partir de $n_{t'}$, es decir, romper los caminos en los arcos anteriores a n_t y $n_{t'}$ y reconectar la primera parte de cada uno con la segunda del otro (ver Figura 3.2.1c).

Mov. 4: insertar una tarea. Para $c \in C$ y $t \in T$, insertar el nodo n_t en r_c (ver Figura 3.2.1d).

Mov. 5: remover una tarea. Para $c \in C$ y $t \in T$, remover el nodo n_t de r_c (ver Figura 3.2.1e).

La vecindad generada por Mov. i, con $i \in \{1, \ldots, 5\}$, notada $N_i(sol)$, contiene a las soluciones factibles que se obtienen de aplicar una vez este movimiento a la solución actual de todas las formas posibles. Recordar que para que la nueva solución sol' siga siendo factible, debe verificar las siguientes condiciones: (i) r'_c es un (n_{l_c}, n_s)-camino dirigido de G_2^δ para cada $c \in C$, (ii) n_t está en al menos uno y en a lo sumo dos caminos dirigidos para cada $t \in T$ y (iii) cada conductor cumple con la legislación laboral. En particular, la condición (ii) implica que Mov. 4 solo puede aplicarse con tareas que se encuentren en una única ruta y Mov. 5 con tareas que se encuentren en exactamente dos rutas.

Algunos de los movimientos se podrían obtener como combinación de otros. Por ejemplo, Mov. 2 se podría obtener aplicando dos veces Mov. 1: primero reubicar a n_t entre u' y $n_{t'}$ y luego reubicar a $n_{t'}$ entre u y v. Sin embargo, existe la posibilidad de que la solución intermedia no sea factible y que la solución final si lo sea. Para evitar lidiar con estos casos, se decidió considerar cada uno de estos movimientos por separado a pesar de que algunos no sean minimales.

El pseudocódigo de la búsqueda local se muestra en el Algoritmo 6. El parámetro *descenso* determina el tipo de descenso, pudiendo elegir entre descender a un vecino de costo mínimo ("MEJOR") o al primer vecino de menor costo ("PRIMER"). El costo de la función objetivo tiende a saltar de manera más abrupta cuando la búsqueda desciende al mejor vecino, a costa de una exploración exhaustiva del vecindario, y ocurre lo contrario al descender al primer vecino. El parámetro *ordenMov* es un arreglo con los movimientos ordenados según su orden de apli-

cación. Por ejemplo, $ordenMov = [2, 5, 1, 4, 3]$ establece que MOV. 2 es el primer movimiento a aplicar, luego MOV. 5, y así sucesivamente.

Algoritmo 6 `búsquedaLocal`

Entrada: *sol* : solución.
Salida: *sol*: solución.
Parámetros: *descenso* $\in$ {"PRIMER", "MEJOR"}, *ordenMov* : arreglo que contiene una permutación de $\{1, \ldots, 5\}$.

```
1:  seguir ← verdadero
2:  while seguir and f_TAXI(sol) > 0 do
3:      seguir ← falso
4:      i ← 0
5:      while i < 5 do
6:          k ← ordenMov[i]
7:          sol* ← sol
8:          for all sol' ∈ N_k(sol) do
9:              if f_TAXI(sol') < f_TAXI(sol*) then
10:                 sol* ← sol'
11:                 if descenso = "PRIMER" then break
12:         if f_TAXI(sol*) < f_TAXI(sol) then
13:             seguir ← verdadero
14:             sol ← sol*
15:         else
16:             i ← i + 1
17: return sol
```

El bucle que comienza en la línea 2 se ejecuta siempre que el valor objetivo de la solución actual sea mayor que 0 y que en la iteración anterior se haya mejorado su valor objetivo (o esta sea la primera iteración). El bucle que comienza en la línea 5 explora secuencialmente las estructuras de vecindario. En la línea 6, se determina el movimiento MOV. k que se debe aplicar según el orden definido por el parámetro *ordenMov*. En la línea 7, se guarda una copia de la solución actual en la variable sol^*, que almacenará la solución a la que debe descender la búsqueda. El bucle que comienza en la línea 8 explora iterativamente cada uno de los vecinos de *sol* generados por MOV. k y actualiza sol^* cuando encuentra un vecino de mejor valor objetivo. En el caso de que el descenso sea al primer vecino, el bucle corta anticipadamente cuando un mejor vecino es encontrado (puede no existir), y en caso de descender al mejor vecino, el bucle recorre el vecindario completo. En las líneas 12–14, la búsqueda se mueve al vecino guardado en sol^*, siempre que sol^* haya sido actualizado. De lo contrario, en la línea 16, se pasa a la siguiente estructura de vecindario. El procedimiento se detiene tras explorar

todas las estructuras de vecindario y en ninguna de ellas haber hallado soluciones mejores.

A continuación, se proponen algunas mejoras a este esquema base, a fin de mejorar algunos comportamientos observados durante pruebas preliminares.

3.3. GRASP con reparación

Uno de los principales puntos débiles del Algoritmo 5, es que la construcción de la solución inicial falla cuando una tarea no puede ser insertada en la ruta de ningún conductor. Esto sucede cuando los conductores se encuentran ocupados durante ese instante de tiempo, o cuando están disponibles pero la inserción de la nueva tarea viola alguno de sus descansos. En pruebas preliminares, se detectó que estos fallos son muy comunes y que la mayoría de ellos provienen de la segunda causa, particularmente de lo restrictivo que resulta imponer los descansos de al menos 12 horas durante la heurística constructiva.

En lugar de descartar estas soluciones incompletas, es posible continuar su construcción asignando estas tareas en conflicto al conductor que más cerca esté de poder realizarlas. Es decir, se permite que algunas de las restricciones puedan ser violadas durante la construcción, comúnmente llamadas restricciones *flexibles*, mientras las demás continúan impuestas, pero se penaliza la infactibilidad con una función objetivo apropiada. Posteriormente, estas soluciones se intentan reparar antes de iniciar la fase de mejora. Para formalizar esto, se introduce la siguiente definición.

Definición 3.3.1. Una *solución semifactible* es una solución factible del problema relajado que se obtiene de eliminar las restricciones que imponen a los conductores un descanso de al menos 12 horas en todo intervalo de 24 horas.

A continuación, se detalla la métrica utilizada para medir cuán infactible es una solución semifactible.

3.3.1. Métrica de infactibilidad

Se propone una función, notada f_{INF}, como métrica de infactibilidad, que retorna el total de horas de trabajo que los conductores exceden de las permitidas en todos los intervalos de 24 horas del horizonte de planificación. Esto es, dada una solución semifactible $sol = (\delta, \{r_c : c \in C\})$,

$$\begin{aligned} f_{\text{INF}}(r_c) &= \sum_{i=0}^{24H-24} \text{máx}\{0, \texttt{horasTrabajadas}(r_c, i, i+24) - 12\} \\ f_{\text{INF}}(sol) &= \sum_{c \in C} f_{\text{INF}}(r_c), \end{aligned} \tag{3.4}$$

donde $\texttt{horasTrabajadas}(r_c, i, i+24)$ es una función que devuelve la cantidad de horas que el conductor trabaja en el intervalo de tiempo $[i, i+24]$ en la ruta r_c.

Es importante destacar que, dada una solución semifactible sol, $f_{\text{INF}}(sol) = 0$ si y sólo si sol es factible. En consecuencia, una heurística de reparación puede usar a f_{INF} como función objetivo para guiar la búsqueda de soluciones factibles. Esta métrica de infactibilidad además tiene otra ventaja, pues penaliza en mayor medida a los conductores que sistemáticamente hacen pocas horas extras sobre aquellos que hacen esporádicamente muchas horas extras. Reparar tales soluciones es más costoso debido a que involucra modificar la planificación de múltiples días; como lo ilustra el siguiente ejemplo.

Ejemplo 3.3.2. Considerar una instancia con un horizonte de planificación de 12 días y dos conductores c_1 y c_2. Suponer que en el primer día, c_1 trabaja las 24 horas, y en los días restantes descansa las primeras 12 horas y luego trabaja las 12 horas restantes, y suponer que c_2 trabaja las primeras 13 horas de cada día y descansa las 11 horas restantes. Si bien ambos conductores trabajan en total 12 horas extras, c_1 aporta 78 horas a f_{INF} (el primer intervalo de 24 horas contribuye con 12 horas, el segundo con 11 horas, y así sucesivamente, y del treceavo en adelante aportan 0), mientras que c_2 aporta 265 horas (cada intervalo de 24 horas contribuye con 1 hora).

En lo que sigue, se describe una heurística para la construcción de soluciones semifactibles.

3.3.2. Heurística de construcción semifactible

La construcción de soluciones semifactibles se realiza con una heurística, llamada `construirSF`, que incorpora ligeros cambios al Algoritmo 5. En lugar de presentar su pseudocódigo, se enumeran los cambios necesarios.

1. Si LC es vacía (línea 5), en lugar de fallar, se construye una lista de candidatos alternativa LC_{INF} con todos los conductores disponibles para cumplir la nueva tarea y que con su incorporación tengan al menos un día de franco en todo intervalo de 7 días (ya no se les exige que descansen al menos 12 horas en todo intervalo de 24 horas). Es decir,

$$LC_{\text{INF}} \leftarrow \{c \in C : r_c \cup \{n_t\} \text{ es un camino dirigido de } G_2^\delta \\ \wedge \ c \text{ cumple restricciones de franco en 7 días}\}.$$

2. Si LC_{INF} no es vacía, se elige un conductor c^* cuya ruta introduzca la menor infactibilidad, de acuerdo a la función f_{INF}. Es decir,

$$c^* \leftarrow \text{argmin}_{c \in LC_{\text{INF}}}(f_{\text{INF}}(r_c \cup \{n_t\}) - f_{\text{INF}}(r_c))$$

3. De lo contrario, la heurística falla y una nueva iteración de GRASP comienza.

En caso de que la construcción retorne una solución semifactible que ya sea factible, la fase de búsqueda local puede comenzar inmediatamente. De lo contrario, se requiere repararla para poder continuar.

3.3.3. Heurística de reparación

La reparación de soluciones semifactibles se hace de forma heurística aplicando una búsqueda local, llamada `reparar`, muy similar a la presentada en el Algoritmo 6. Dado que en este caso el objetivo es hallar una solución factible, el descenso está guiado por la función f_{INF} en lugar de f_{TAXI}. A su vez, todos los movimientos definidos en la Sección 3.2.2 son reutilizados para la reparación y se incorpora uno nuevo, capaz de modificar la programación horaria de las tareas.

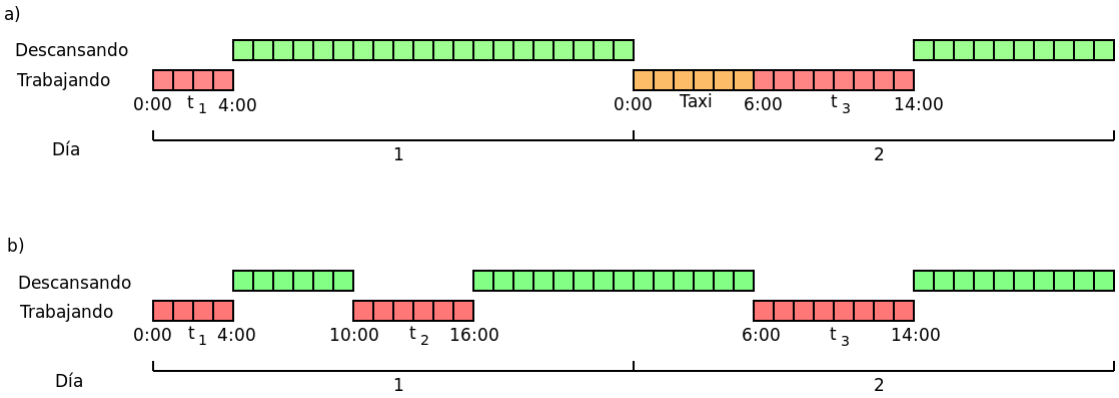


Figura 3.3.1.: Ejemplo de reparación con Mov. 4. a) No factible, b) factible.

Es importante notar que todos los movimientos pueden ser aplicados para reducir la infactibilidad de una solución. A continuación, se muestra un ejemplo de cómo el movimiento Mov. 4, encargado de insertar una tarea en una ruta, puede ser usado para esto. Ejemplos similares pueden construirse para el resto de los movimientos.

Ejemplo 3.3.3. Considerar dos localidades $l_1, l_2 \in L$ tales que $\text{tVT}(l_1, l_2) = 6$, tareas $t_1, t_2, t_3 \in T$ tales que $\text{des}(t_1) = \text{ori}(t_2) = l_1$, $\text{des}(t_2) = \text{ori}(t_3) = l_2$, $\text{dur}(t_1) = 4$, $\text{dur}(t_2) = 6$, $\text{dur}(t_3) = 8$ y una programación horaria δ con $\delta(t_1) = 0$, $\delta(t_2) = 10$ y $\delta(t_3) = 30$. Además, suponer que un mismo conductor realiza t_1 y luego t_3. Dado que tal secuencia de tareas requiere un viaje en taxi intermedio desde l_1 hacia l_2 y que, de acuerdo a lo asumido, el mismo se programa inmediatamente antes de t_3, tal conductor trabaja 14 horas el intervalo $[24, 48]$, ver Figura 3.3.1a). Ahora, si en su ruta se inserta t_2 entre t_1 y t_3, entonces el taxi ya no es necesario y la infactibilidad se repara, ver Figura 3.3.1b).

Como ha sido mencionado anteriormente, ninguno de los movimientos presentados puede modificar la programación horaria de las tareas. Considerando que una solución necesita ser reparada cuando algún conductor excede las 12 horas de trabajo estipuladas, se vuelve deseable tener un movimiento que pueda adelantar/demorar el horario de una tarea para intentar incrementar el tiempo de descanso del conductor en ese intervalo. El nuevo movimiento se presenta a continuación, su entrada es una solución semifactible $sol = (\delta, \{r_c : c \in C\})$ y retorna otra solución $sol' = (\delta', \{r_c : c \in C\})$.

Mov. 6: cambiar el horario de una tarea. Para $t \in T$ y un instante de tiempo tiempo $i \in \mathcal{I}_t$, actualizar el valor de $\delta(t)$ a i.

Las siguientes tres condiciones se deben cumplir para que sol' sea semifactible.

1. δ' debe ser nuevamente una programación horaria, de acuerdo a la Definición 2.1.6. Es decir, si t_1 y t_2 son las tareas que preceden y siguen a t en la ruta del camión que realiza t, respectivamente, entonces se debe cumplir

$$\delta(t_1) + \text{dur}(t_1) \leq i \leq \delta(t_2) - \text{dur}(t).$$

2. r_c debe ser un (n_{l_c}, n_s)-camino dirigido de $G_2^{\delta'}$, para cada $c \in C$. Es decir, si t_1' y t_2' son las tareas que preceden y siguen a t en la ruta de un conductor que realiza t (hay al menos uno y a lo sumo dos de estos conductores), respectivamente, entonces se debe cumplir

$$\begin{aligned} i &\geq \delta(t_1') + \text{dur}(t_1') + \text{tiempoViajeTaxi}(\text{des}(t_1'), \text{ori}(t)), \\ i &\leq \delta(t_2') - \text{dur}(t) - \text{tiempoViajeTaxi}(\text{des}(t), \text{ori}(t_2')). \end{aligned}$$

   Pero si t era la primera tarea de un conductor c, entonces se debe cumplir

$$i \geq \text{tiempoViajeTaxi}(l_c, \text{ori}(t)).$$

3. c tiene al menos un día de franco en todo intervalo de 7 días, para cada $c \in C$.

En lugar de presentar el pseudocódigo de la heurística de reparación, que es muy similar al Algoritmo 6, se enumeran los cambios necesarios.

1. El descenso está guiado por la función f_{INF} en lugar de f_{TAXI} (líneas 2, 9, y 13).

2. Los parámetros se renombran a $descensoR$ y $ordenMovR$. Además, $ordenMovR$ contiene ahora una permutación de $\{1, \ldots, 6\}$.

3. La condición del bucle while de la línea 6 cambia a $i < 6$.

4. La condición del bucle while de la línea 8 cambia a $sol' \in N_k^{\text{SF}}(sol)$, donde $N_k^{\text{SF}}(sol)$ es el conjunto de soluciones semifactibles que se obtienen de aplicar una vez MOV. k a sol de todas las formas posibles.

Es de esperar que al usar únicamente f_{INF} para guiar el descenso, la búsqueda local descienda rápidamente hacia soluciones factibles, aunque las mismas pueden ser de mala calidad (respecto a f_{TAXI}) para arrancar la fase de mejora. Otra alternativa es guiar la reparación con una suma ponderada entre f_{INF} y f_{TAXI}. Tal enfoque, requeriría normalizar ambas funciones y experimentar para encontrar los pesos más convenientes para la suma ponderada. Esta alternativa se deja planteada como trabajo futuro.

3.3.4. Esquema GRASP+R

El esquema GRASP+R incorpora a GRASP las mejoras mencionadas recientemente. Durante la fase de construcción, se aplica la heurística de construcción semifactible y se invoca una heurística de reparación antes de arrancar la fase de mejora. Como es esperable que soluciones muy infactibles no logren ser reparadas o que su reparación tome demasiado tiempo, se incorpora a este esquema un umbral de reparación adaptativo, el cual decide si vale la pena reparar una solución en base a su propio valor objetivo y al de las soluciones semifactibles que pudieron ser reparadas con éxito anteriormente. El resultado se muestra en el Algoritmo 7.

En la línea 1, se fija el valor del umbral a $+\infty$ y, en las líneas 8–9, se reemplaza por el valor objetivo (respecto a f_{INF}) de la primera solución semifactible (pero no factible) que logre ser reparada exitosamente. De esta forma, inicialmente se reparan todas las soluciones iniciales, hasta encontrar la primera solución factible, y allí el umbral adquiere de forma automática un valor inicial. En las líneas 4–5, se intenta construir una solución semifactible inicial y, en caso de que la construcción falle, se sigue con la siguiente iteración. En las líneas 6–7, se intenta reparar la solución siempre que el valor de su infactibilidad sea menor o igual al establecido por el umbral de reparación. Ya que el valor inicial del umbral puede no ser representativo (por ejemplo, la heurística pudo haber construido de forma fortuita una muy buena solución que fije el umbral en un valor muy bajo, haciéndolo muy restrictivo para las futuras iteraciones), en la línea 10, se ajusta su valor considerando el valor

Algoritmo 7 `GRASP+R`

Entrada: T: conjunto de tareas y $\delta : T \to \mathcal{I}$ programación horaria.
Salida: sol^*: solución factible.
Parámetros: $t_{lim} \in \mathbb{R}_+$.

```
 1: umbralR ← +∞
 2: w* ← +∞
 3: while no se excede t_lim do
 4:     sol ← construirSF(T, δ)
 5:     if hay fallo then continuar
 6:     if f_INF(sol) ≤ umbralR then
 7:         sol' ← reparar(sol)
 8:         if umbralR = +∞ ∧ f_INF(sol) > 0 ∧ f_INF(sol') = 0 then
 9:             umbralR ← f_INF(sol)
10:     umbralR ← ajustarUmbral(umbralR, f_INF(sol))
11:     if f_INF(sol') > 0 then continue
12:     sol' ← búsquedaLocal(sol')
13:     if f_TAXI(sol') < w* then
14:         w* ← f_TAXI(sol')
15:         sol* ← sol'
16: return sol*
```

objetivo de la solución inicial antes de ser reparada. En la línea 11, se evalúa si la solución reparada es no factible, y de ser así, se descarta y se sigue con la siguiente iteración. De lo contrario, en las líneas 12–15, se da inicio a la fase de mejora y luego se actualiza sol^* en caso de encontrar una mejor solución.

El ajuste del umbral de reparación es realizado por la función `ajustarUmbral`, cuyo pseudocódigo se muestra en el Algoritmo 8. La misma tiene un parámetro $t_{reparacion} \in (0, 1)$, llamado *tasa de ajuste*, que controla la proporción con la que se ajusta el valor del umbral. Valores de $t_{reparacion}$ cercanos a 1, hacen que el valor del umbral salte de manera más abrupta, y lo contrario ocurre para valores cercanos a 0.

En las líneas 2–3, se evalúa si el valor objetivo de la solución inicial excede el del umbral, y de ser así, se incrementa el valor del umbral según la tasa de ajuste. De esta forma, se vuelve más probable que la heurística constructiva genere en las próximas iteraciones soluciones cuyos valores objetivo caigan dentro del umbral y sean reparadas. De lo contrario, en las líneas 4–5, se reduce el valor del umbral de acuerdo a esa tasa. Así, la reparación se vuelve más restrictiva en las próximas iteraciones.

Algoritmo 8 `ajustarUmbral`

Entrada: $umbralR \in \mathbb{R}_+$ y $w \in \mathbb{R}_+$.
Salida: $umbralR' \in \mathbb{R}_+$.
Parámetros: $t_{reparacion} \in (0, 1)$.

```
1: if umbralR < +∞ then
2:     if w > umbralR then
3:         umbralR' ← umbralR * (1 + t_reparacion)
4:     else if w ≤ umbralR then
5:         umbralR' ← umbralR * (1 − t_reparacion)
6: return umbralR'
```

3.4. GRASP con perturbación

Una particularidad de GRASP es que la construcción de la solución inicial no está influenciada por la información reunida en iteraciones anteriores. Esto suele ser señalado como una desventaja frente a otras metaheurísticas, como por ejemplo, búsqueda tabú [Gendreau and Potvin, 2019] o algoritmos genéticos [Reeves, 2010]. En esta sección, se propone aplicar una combinación de GRASP con otra metaheurística, dando lugar a un GRASP híbrido, para compensar su falta de memoria.

La elegida para la combinación es *Iterated Local Search* (ILS) [Baxter, 1981, Lourenço et al., 2019]. Esta decisión se debe a dos motivos: por un lado, es una metaheurística que ha resultado muy competitiva en la resolución de problemas de optimización, por ejemplo en VRPs [Hashimoto et al., 2008, Laurent and Hao, 2009] y en especial para el *Traveling Salesman Problem* [Merz and Huhse, 2008]; por el otro lado, su incorporación a GRASP es casi directa y puede aprovechar los movimientos ya implementados. El esquema general de ILS se describe en el Apéndice 4.3.

El resultado de esta hibridación es con frecuencia denotada en la literatura como GRASP×ILS (e.g. en [Ait Haddadene et al., 2016]). Este esquema híbrido busca aprovechar el esfuerzo computacional requerido para encontrar un óptimo local antes de descartarlo para dar comienzo a la siguiente iteración de GRASP. Para ello, se aplica sobre el óptimo local una heurística de perturbación para intentar desatascar la búsqueda local. Por supuesto, existe un compromiso, pues una perturbación muy sutil difícilmente libere el atasco y cambios muy grandes proba-

blemente desechen todas las mejoras logradas, lo que sería similar a comenzar la búsqueda local desde una nueva solución inicial.

El Algoritmo 9 muestra el pseudocódigo del GRASP×ILS desarrollado, el cual incorpora un par de cambios sobre el Algoritmo 7. Además del tiempo límite, toma como parámetro de entrada un natural $maxPerturb$ que establece la máxima cantidad de perturbaciones que se aplicarán a cada óptimo local. Se agregan dos variables $nIter$ y $nFallos$ para contabilizar el número de iteración actual y el número de iteraciones fallidas (en las que no pudo alcanzarse una solución factible). En las líneas 20–22, la solución es repetidamente perturbada y mejorada. El número de perturbaciones aplicadas es adaptativo, tendiendo a 0 cuando la proporción de iteraciones fallidas es baja y a $maxPerturb$ en caso contrario. De esta forma, si la búsqueda de soluciones factibles se torna dificultosa, las pocas soluciones encontradas son aprovechadas lo máximo posible antes de ser descartadas. A diferencia de lo mostrado en el apéndice, nuestra implementación aplica la perturbación directamente sobre la solución y no sobre una copia, debido a que la misma no modifica su valor objetivo respecto a f_{TAXI}, tal como se verá a continuación.

Resta definir la heurística de perturbación. Una forma popular y poco costosa de hacerlo es aplicando de forma aleatoria algunos de los movimientos considerados en la búsqueda local. En este caso, el movimiento MOV. 6 es particularmente útil porque tiene la propiedad de que su aplicación no modifica el costo de la solución respecto a la distancia de viaje en taxi (los conductores necesitan los mismos viajes en taxi, aunque quizás programados a un horario diferente). De esta forma, si la solución devuelta por MOV. 6 es factible, la misma coincide en valor objetivo con la original y con suerte ofrece una oportunidad de escape en la estructura de vecindario de algún otro movimiento.

El Algoritmo 10 muestra la heurística de perturbación desarrollada. El bucle que comienza en la línea 1 recorre todas las tareas en un orden aleatorio e intenta cambiar el horario programado para cada tarea de forma también aleatoria. El conjunto $N_6(sol, t)$ está formado por todas las soluciones factibles que se obtienen de aplicar MOV. 6 a sol para cambiar el horario de t de todas las formas posibles. Es decir, la única diferencia entre sol y una solución de $N_6(sol, t)$ es el horario programado para t. Si este conjunto es vacío, en la línea 2, se continúa con la siguiente tarea, sin modificar su horario. De lo contrario, se elige un vecino aleatorio

Algoritmo 9 `GRASP×ILS`

Entrada: T: conjunto de tareas y $\delta : T \to \mathcal{I}$ programación horaria.
Salida: sol^*: solución factible.
Parámetros: $t_{lim} \in \mathbb{R}_+$ y $maxPerturb \in \mathbb{N}_0$.
Salida: A feasible solution S^*

1: $umbralR \leftarrow +\infty$
2: $w^* \leftarrow +\infty$
3: $nIter \leftarrow 0$
4: $nFallo \leftarrow 0$
5: **while** no se excede t_{lim} **do**
6: $\quad nIter \leftarrow nIter + 1$
7: $\quad sol \leftarrow$ `construirSF`(T, δ)
8: $\quad$ **if** hay fallo **then**
9: $\quad\quad nFallo \leftarrow nFallo + 1$
10: $\quad\quad$ **continue**
11: $\quad$ **if** $f_{\text{INF}}(sol) \leq umbralR$ **then**
12: $\quad\quad sol' \leftarrow$ `reparar`(sol)
13: $\quad\quad$ **if** $umbralR = \infty \wedge f_{\text{INF}}(sol) > 0 \wedge f_{\text{INF}}(sol') = 0$ **then**
14: $\quad\quad\quad umbralR \leftarrow f_{\text{INF}}(sol)$
15: $\quad umbralR \leftarrow$ `ajustarUmbral`$(umbralR, f_{\text{INF}}(sol))$
16: $\quad$ **if** $f_{\text{INF}}(sol') > 0$ **then**
17: $\quad\quad nFallo \leftarrow nFallo + 1$
18: $\quad\quad$ **continue**
19: $\quad sol' \leftarrow$ `búsquedaLocal`(sol')
20: $\quad$ **for** $i = 1, \ldots, \lfloor maxPerturb^{(nFallo/nIter)} \rfloor$ **do**
21: $\quad\quad sol' \leftarrow$ `perturbar`(sol')
22: $\quad\quad sol' \leftarrow$ `búsquedaLocal`(sol')
23: $\quad$ **if** $f_{\text{TAXI}}(sol') < w^*$ **then**
24: $\quad\quad w^* \leftarrow f_{\text{TAXI}}(sol')$
25: $\quad\quad sol^* \leftarrow sol'$
26: **return** sol^*

para reemplazar a la solución actual, y se procede con la siguiente tarea.

Algoritmo 10 `perturbar`

Entrada: *sol* solución factible.
Salida: *sol* : solución factible.

1: **for all** $t \in T$ **do**
2: **if** $N_6(sol, t) = \emptyset$ **then continuar**
3: $sol \leftarrow rand(N_6(sol, t))$
4: **return** *sol*

4. Experiencias computacionales

Este capítulo se enfoca en evaluar los algoritmos propuestos para la resolución heurística de cada uno de los dos subproblemas secuenciales en que se descompone el problema en estudio, sobre un conjunto de instancias de prueba generadas de forma pseudoaleatoria con hasta 3000 pedidos, 520 camiones, 1040 conductores y un horizonte de planificación de hasta 28 días. Previo a esto, se realiza un extenso ajuste de los parámetros involucrados en los algoritmos a fin de hallar la configuración de valores más recomendable. Cada una de las experiencias computacionales llevadas a cabo se acompaña de cuadros para mostrar los resultados obtenidos y también se plantea una discusión sobre los mismos.

Se dispone de una computadora equipada con un procesador Intel i5 2.66GHz (usando un único thread), 5.7GB de memoria, sistema operativo Ubuntu 20.04 64bits. Todas las implementaciones fueron escritas en el lenguaje de programación `C++11` y compilado con GCC 9.3.0.

4.1. Generación de instancias

En esta sección, se describen las instancias utilizadas para probar los algoritmos de optimización propuestos. Uno de los principales inconvenientes en el desarrollo de este trabajo fue la imposibilidad de acceder a datos internos de la compañía, como por ejemplo, las localidades involucradas, el volumen de pedidos, el tamaño de la flota de camiones y de conductores, los costos operativos, entre otros. Por este motivo, se decide generar instancias propias de forma pseudoaleatoria, respetando la naturaleza del problema. Para simplificar la redacción, el tiempo se considera discretizado en horas, con $I = 24$, pero es posible usar un valor diferente.

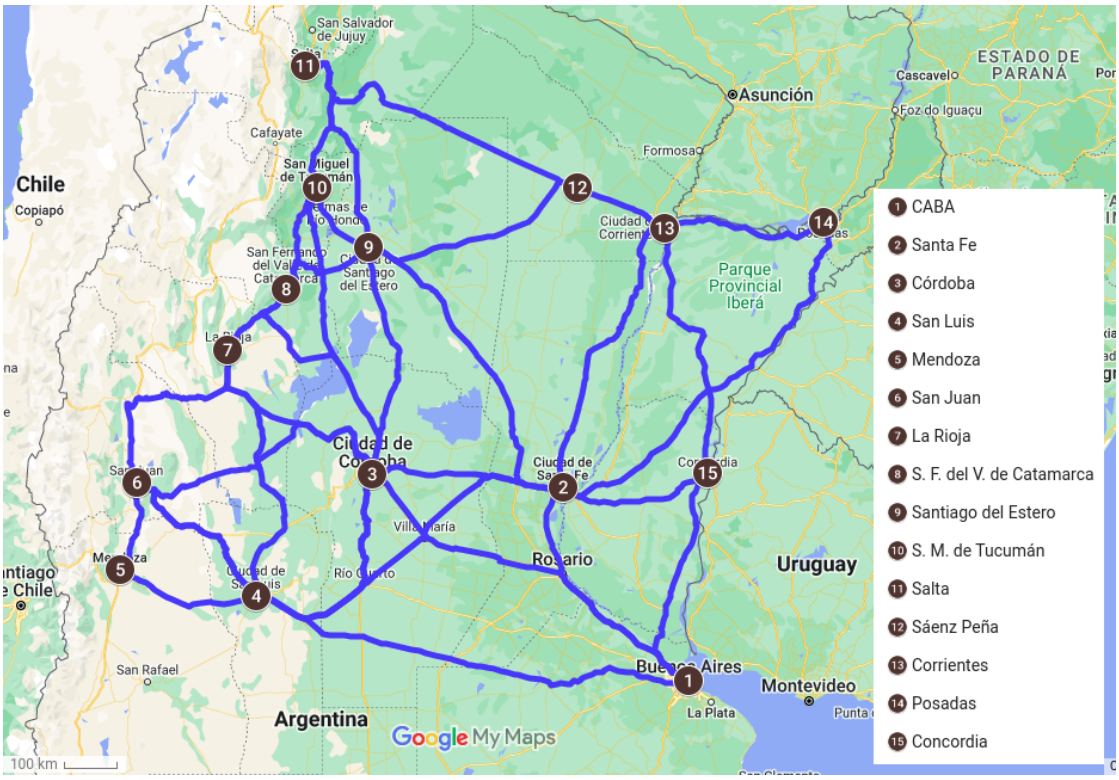


Figura 4.1.1.: Mapa con las 15 ciudades y las rutas consideradas, extraído de *Google Maps*.

En la literatura, es usual simular las localidades como puntos distribuidos aleatoriamente en el plano con sus distancias euclidianas; véase por ejemplo las instancias generadas en [Solomon, 1987] para el VRPTW. Este enfoque clásico fue descartado en esta investigación, debido a que el mapa resultante es poco real pues ignora la geografía del terreno. Por el contrario, se decide utilizar ciudades y distancias reales. En particular, se trabaja con las 15 ciudades argentinas y con las rutas nacionales mostradas en el mapa de la Figura 4.1.1. La mayoría de las ciudades son capitales de su provincia, salvo Concordia (Entre Ríos) y Sáenz Peña (Chaco), que se prefirieron sobre Paraná y Resistencia por la cercanía de estas últimas con Santa Fe y Corrientes, respectivamente. El motivo por el cual no se consideró a San Salvador de Jujuy y Formosa es que forman vértices colgantes, es decir, cada una de ellas es adyacente a una única ciudad. Tampoco se consideraron ciudades de la Patagonia debido a que las distancias entre ellas son demasiado grandes, lo que obligaría a adicionar muchas ciudades de paso para que los conductores pueden descansar (el problema prohíbe descansar en el medio de una ruta).

	CABA	Concordia	Córdoba	Corrientes	La Rioja	Mendoza	Posadas	Sáenz Peña	Salta	Catamarca	San Juan	San Luis	Tucumán	Santa Fe	S. del Estero
CABA	0	427	698	–	–	–	–	–	–	–	–	813	–	470	–
Concordia	427	0	–	503	–	–	587	–	–	–	–	–	–	293	–
Córdoba	698	–	0	–	442	–	–	–	–	452	585	428	582	364	445
Corrientes	–	503	–	0	–	–	319	181	–	–	–	–	–	565	–
La Rioja	–	–	442	–	0	–	–	–	–	157	444	522	–	–	–
Mendoza	–	–	–	–	–	0	–	–	–	–	176	258	–	–	–
Posadas	–	587	–	319	–	–	0	–	–	–	–	–	–	780	–
Sáenz Peña	–	–	–	181	–	–	–	0	656	–	–	–	–	–	444
Salta	–	–	–	–	–	–	–	656	0	–	–	–	307	–	438
Catamarca	–	–	452	–	157	–	–	–	–	0	–	–	234	–	234
San Juan	–	–	585	–	444	176	–	–	–	–	0	325	–	–	–
San Luis	813	–	428	–	522	258	–	–	–	–	325	0	–	653	–
Tucumán	–	–	582	–	–	–	–	–	307	234	–	–	0	–	161
Santa Fe	470	293	364	565	–	–	780	–	–	–	–	653	–	0	611
S. del Estero	–	–	445	–	–	–	–	444	438	234	–	–	161	611	0

Cuadro 4.1.1.: Distancia entre ciudades en km.

En el Cuadro 4.1.1, se resume la distancia en kilómetros entre las ciudades; el símbolo "–" se reporta cuando no existe una ruta directa entre ellas. En el Cuadro 4.1.2, se resume el tiempo de viaje en camión y en taxi entre las ciudades, es decir, las funciones tiempoViajeCamion y tiempoViajeTaxi, respectivamente. Se asume que los vehículos se desplazan a una velocidad constante de 90km/h y los tiempos de viaje con valores decimales se redondean al entero más cercano. Por supuesto, todos los tiempos de viaje tienen una duración menor a 12 horas, de lo contrario no podrían ser realizados por un conductor. Dadas dos ciudades adyacentes l_1 y l_2, se proponen los siguientes costos de viaje costoViajeCamion(l_1, l_2) = tiempoViajeCamion(l_1, l_2) y costoViajeTaxi$(l_1, l_2) = 2 *$ costoViajeCamion(l_1, l_2), dado que el taxi debe regresar a la localidad de origen luego de llevar al conductor a su destino.

Además del conjunto de ciudades L, los tiempos y los costos de viaje, la rutina de generación de instancias recibe como parámetros de entrada al número de días del horizonte de planificación (H), al número de pedidos ($|P|$), al número de camiones ($|V|$) y al número de conductores ($|C|$). Con todos estos datos de entrada realiza lo siguiente:

	CABA	Concordia	Córdoba	Corrientes	La Rioja	Mendoza	Posadas	Sáenz Peña	Salta	Catamarca	San Juan	San Luis	Tucumán	Santa Fe	S. del Estero
CABA	0	5	8	–	–	–	–	–	–	–	–	9	–	5	–
Concordia	5	0	–	6	–	–	7	–	–	–	–	–	–	3	–
Córdoba	8	–	0	–	5	–	–	–	–	5	7	5	6	4	5
Corrientes	–	6	–	0	–	–	4	2	–	–	–	–	–	6	–
La Rioja	–	–	5	–	0	–	–	–	–	2	5	6	–	–	–
Mendoza	–	–	–	–	–	0	–	–	–	–	2	3	–	–	–
Posadas	–	7	–	4	–	–	0	–	–	–	–	–	–	9	–
Sáenz Peña	–	–	–	2	–	–	–	0	7	–	–	–	–	–	5
Salta	–	–	–	–	–	–	–	7	0	–	–	–	3	–	5
Catamarca	–	–	5	–	2	–	–	–	–	0	–	–	3	–	3
San Juan	–	–	7	–	5	2	–	–	–	–	0	4	–	–	–
San Luis	9	–	5	–	6	3	–	–	–	–	4	0	–	7	–
Tucumán	–	–	6	–	–	–	–	–	3	3	–	–	0	–	2
Santa Fe	5	3	4	6	–	–	9	–	–	–	–	7	–	0	7
S. del Estero	–	–	5	–	–	–	–	5	5	3	–	–	2	7	0

Cuadro 4.1.2.: Tiempo de viaje entre ciudades en horas.

1. Genera los pedidos, eligiendo los atributos de cada pedido p de la siguiente forma:

   1.1. l_p^{R} y l_p^{E} son ciudades aleatorias de L, distintas entre sí.
   1.2. a_p^{R} y b_p^{R} son enteros aleatorios entre 0 y 23, distintos entre sí. Análogamente para a_p^{E} y b_p^{E}.
   1.3. dia_p^{R} y dia_p^{E} son enteros aleatorios entre 0 y $H-1$, tales que $dia_p^{\mathrm{R}} \leq dia_p^{\mathrm{E}}$.
   1.4. $s_p^{\mathrm{R}} = s_p^{\mathrm{E}} = 1$.
   1.5. w_p es un entero aleatorio en $[5, 15]$.

2. Para cada vehículo v, elige una ciudad inicial l_v aleatoria en L y con probabilidad 0.8 ubica también a un conductor en esa ciudad inicial.

3. Para cada conductor c restante (asumiendo $|V| \leq |D|$), elige una ciudad inicial l_c aleatoria en L.

Con esta generación, es posible que existan ciudades que tengan inicialmente más camiones que conductores, y viceversa, pero ambos escenarios son consistentes con la naturaleza del problema. Por ejemplo, entre una planificación y la siguiente,

la planta de conductores pudo haberse modificado o los conductores pudieron haberse vuelto a sus hogares. Otro ejemplo donde los camiones y los conductores se desparejan ocurre cuando un camión con dos conductores deja uno de ellos descansando en una ciudad y el otro continúa la conducción. Con la ayuda de esta rutina, se construyeron dos conjuntos de instancias de prueba, llamados `Set1.1` y `Set1.2`; a continuación se brindan más detalles sobre la construcción de los mismos.

- `Set1.1`. Se genera una instancia con $|P| = 100$ para cada combinación de $H = 7$ y $|V| \in \{14, 21, 28\}$, esto se repite para $H = 14$ y $|V| \in \{10, 15, 20\}$, también para $H = 28$ y $|V| \in \{6, 9, 12\}$. El número de conductores depende del número de camiones, más precisamente, $|C| = 3,5|V|$ para el número más ajustado de camiones, $|C| = 2,5|V|$ para el intermedio y $|C| = 2|V|$ para el más alto. Para una mayor aleatoriedad en la generación, tras cada llamada a la rutina se cambia la semilla usada en la generación de enteros aleatorios. Así, se obtienen 9 instancias distintas.

- `Set1.2`. Este conjunto está conformado por instancias mucho más grandes que las de `Set1.1`. Se genera una instancia con $|P| = 1000$ para cada una de las siguientes configuraciones: $\{H = 7, |V| = 160\}$, $\{H = 14, |V| = 120\}$ y $\{H = 28, |V| = 120\}$, en todas ellas se usa $|C| = 2,25|V|$. Análogamente, se genera una instancia con $|P| = 3000$ para cada una de las siguientes configuraciones: $\{H = 7, |V| = 520\}$, $\{H = 14, |V| = 410\}$ y $\{H = 28, |V| = 300\}$, en todas ellas se usa $|C| = 2|V|$. Luego, para cada una de las 6 instancias generadas, se crea una copia de la misma y se le incorporan $0,25|V|$ nuevos conductores con localidades iniciales aleatorias. En total, se tienen 12 instancias diferentes.

El valor elegido para $|P| = 100$ es usual en otras instancias benchmark para VRPs; véase [Solomon, 1987]. Los números de camiones y de conductores propuestos fueron determinados de manera experimental, de modo que el problema no sea fácil de resolver ni que sea infactible. Creemos que el número de conductores considerados es razonable, dado que al menos dos conductores son requeridos para usar un camión por más de 12 horas seguidas. También es esperable que la tasa de conductores por camión sea mayor cuanto menor sea la flota de camiones, dado que

los camiones necesitan cubrir mayores distancias y los conductores probablemente necesiten taxis con mayor frecuencia.

Por último, vale la pena mencionar algunas consideraciones adicionales que se incorporaron a la generación para prevenir que las instancias sean muy duras o infactibles.

- Dado que ventanas de tiempo demasiado cortas serían poco realistas, tanto $[a_p^{\mathrm{R}}, b_p^{\mathrm{R}}]$ como $[a_p^{\mathrm{E}}, b_p^{\mathrm{E}}]$ deben abrir por al menos 4 horas consecutivas. Notar que la ventana de tiempo de mayor extensión ocurre cuando $b_p^{\mathrm{R}} + 1 \equiv a_p^{\mathrm{R}}$ mód 24, por ejemplo, $[a_p^{\mathrm{R}}, b_p^{\mathrm{R}}] = [0, 23]$ o $[a_p^{\mathrm{R}}, b_p^{\mathrm{R}}] = [8, 7]$ y en ese caso, la recolección puede comenzar en cualquier instante de tiempo del horizonte a partir de su día inicial.

- Dado que un pedido sería difícil de cumplir si su recolección o entrega comenzase cerca del final del horizonte, se procura que $dia_p^{\mathrm{R}} \leq H-4$ y $dia_p^{\mathrm{E}} \leq H-2$. Recordar que no es necesario que los pedidos sean recolectados ni entregados en su día inicial, aunque las entregas tardías incurren en costos adicionales.

- Para evitar que los pedidos sean necesariamente entregados con demora, se garantiza que dia_p^{E} sea lo suficientemente posterior a dia_p^{R}. Por ejemplo, considerar una instancia con un pedido p con los siguientes atributos: $[a_p^{\mathrm{R}}, b_p^{\mathrm{R}}] = [3, 10]$, $dia_p^{\mathrm{R}} = 0$, $s_p^{\mathrm{R}} = 1$ y $\mathrm{tiempoViajeCamion}(l_p^{\mathrm{R}}, l_p^{\mathrm{E}}) = 5$. Si un camión recolecta p lo más pronto posible, esto es en el instante de tiempo 3 del primer día, entonces su entrega puede comenzar al menos 6 instantes de tiempo más tarde, dado que el camión debe esperar el tiempo de servicio y viajar a la localidad de entrega. En particular, si la ventana de tiempo de entrega ya se encontrase cerrada al llegar a la localidad de entrega, entonces sería necesario esperar al próximo día para entregarlo, lo que conlleva inevitablemente a penalidad por demora. Es decir, si además $[a_p^{\mathrm{E}}, b_p^{\mathrm{E}}] = [2, 8]$ y $dia_p^{\mathrm{E}} = 0$, entonces la entrega necesariamente debe comenzar a partir del segundo día, dado que ningún camión puede recolectar a p y viajar a la localidad de entrega antes del instante de tiempo 8. Formalmente, siendo $dia_p^{\mathrm{E}*}$ el mínimo día en el cual la ventana de tiempo de entrega de p cierra más tarde que el menor instante de tiempo en que un camión pudo haber recolectado a p y viajado a la localidad de entrega (asumiendo que fue recolectado y transportado sin demoras), se garantiza que $dia_p^{\mathrm{E}} \geq dia_p^{\mathrm{E}*}$. Observar que esta

condición es necesaria pero no suficiente para garantizar que existan soluciones factibles sin demoras en las entregas, por ejemplo podría ocurrir que no haya forma de organizar los camiones y conductores para recolectar un pedido en el primer instante posible de tiempo.

4.2. Configuración de parámetros

Los algoritmos de optimización propuestos involucran un gran número de parámetros cuyos valores más adecuados deben ser determinados experimentalmente. Tradicionalmente, esto requeriría evaluar todas las posibles combinaciones de valores sobre todas las instancias y realizar un análisis de los resultados obtenidos para elegir la mejor configuración. El principal problema de este abordaje es el enorme tiempo de cómputo que se precisa. En este trabajo, se propone la utilización de *irace* para la configuración automática de los parámetros. Este paquete utiliza una técnica conocida como *racing* para descartar anticipadamente (antes de resolver todas las instancias) aquellas configuraciones para las cuales se tenga suficiente evidencia estadística de que se comportan peor que al menos otra configuración. Los detalles más técnicos pueden ser consultados en [López-Ibáñez et al., 2016].

Configuración de LNS

La configuración de los parámetros de LNS se realiza incrementalmente para evitar lidiar con todos los parámetros en simultáneo. Esto es, en cada llamada a irace se consideran algunos de los parámetros y se mantienen los valores de los parámetros que ya fueron configurados previamente. Se propone el siguiente orden de parámetros para la configuración.

1. Parámetros involucrados en el criterio de parada (t_{frio} y t_0), el factor de arrepentimiento (k) y la proporción de pedidos removidos (α), usando únicamente el operador de eliminación aleatorio, $nIters = 50000$, y $ruido_{\text{ENTR}} = ruido_{\text{VIAJ}} = 0$.

2. Parámetros de cada una de las eliminaciones de Shaw (λ_{shaw} y e_{shaw}) y de peores (e_{peores}).

Parámetro	Configuraciones	Mejor valor
t_{frio}	0.999, 0.99925, 0.9995, 0.99975, 0.9999	**0.99975**
t_0	0, 0.025, 0.05, 0.075, 0.1	**0.05**
k	2, 3	**2**
α	0.1, 0.2, 0.3, 0.4, 0.5	**0.3**
λ_{shaw}	0, 0.25, 0.5, 0.75, 1	**0.75**
e_{shaw}	1, 1.5, 2, 3, 6	**3**
e_{peores}	1, 1.5, 2, 3, 6	**1.5**
OpDestr	"A", "S", "P"	"P"
	"P", "AP"	**"AP"**
$ruido_{\text{ENTR}}$	0, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1	**0**
$ruido_{\text{VIAJ}}$	0, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1	**0.01**

Cuadro 4.2.1.: Ajuste de parámetros de LNS

3. Se incorpora un nuevo parámetro, $OpDestr \in \{$"A","S","P"$\}$, que establece el operador de destrucción aplicado: aleatorio, Shaw y peores, respectivamente.

4. Dado que las configuraciones $OpDestr =$ "A" y $OpDestr =$ "P" obtienen ambas buenos resultados, se incorpora un nuevo valor para este parámetro, denominado "AP", que elige de forma aleatoria el operador de destrucción a aplicar, entre la eliminación aleatoria y de peores, en cada iteración de LNS.

5. Parámetros de ruido $ruido_{\text{ENTR}}$ y $ruido_{\text{VIAJ}}$.

El Cuadro 4.2.1 muestra las configuraciones analizadas para cada parámetro y su valor más recomendable según irace. El valor final elegido para cada parámetro se encuentra resaltado en negrita. Para este ajuste, se usaron 3 instancias arbitrarias de `Set1`, cada una de ellas variando λ (el parámetro que pondera el peso de las funciones objetivo) en $\{0; 0{,}25; 0{,}5; 0{,}75; 1\}$, lo que da un total de 15 instancias.

Configuración de GRASP

De la misma forma, la configuración de GRASP también se realiza incrementalmente, considerando primero los parámetros exclusivos de GRASP, luego los de

GRASP-R y finalmente los de GRASP×ILS, manteniendo siempre $t_{lim} = 10$ minutos. La única complicación que surge involucra al parámetro $ordenMov$ (y análogamente $ordenMovR$), el cual posee una gran cantidad de valores en su dominio. Por ejemplo, existen 120 permutaciones de 5 movimientos y asciende a 320 si se permite que algunos de los movimientos estén apagados. Siguiendo la idea anterior, se propone un ajuste incremental para $ordenMov$. Primero, se ejecuta irace con algunas pocas permutaciones de interés. En particular, se procura que Mov. 4 y 5 se exploren antes que Mov. 1, 2, y 3, pues VND explora las primeras estructuras de vecindario muchas más veces y aplicar Mov. 4 y 5 requiere un menor esfuerzo computacional. Luego, se vuelve a llamar a irace con las permutaciones ganadoras y las obtenidas de desactivar algunos movimientos de las mismas. Esto se repite hasta que la desactivación de movimientos no logre mejorar los resultados.

El Cuadro 4.2.2 muestra el resumen de los resultados obtenidos. Para realizar este ajuste, se seleccionaron 3 instancias arbitrarias de `Set1`. Sobre cada una de ellas, se aplicó la LNS ya configurada para resolver la primera etapa, variando $\lambda \in \{0; 0{,}25; 0{,}5; 0{,}75; 1\}$, lo que da como resultado un total de 15 instancias para iniciar la segunda etapa. Se usa un arreglo de menor longitud para indicar que algunos movimientos están apagados; por ejemplo, $ordenMov = [4, 5]$ quiere decir que primero se aplica Mov. 4, luego Mov. 5 y que los demás movimientos no se aplican. La notación p$\{x_1, \ldots, x_n\}$ se refiere a una permutación de $\{x_1, \ldots, x_n\}$; por ejemplo, [p{4,5}, p{1,2,3}] representa a todos los arreglos donde los primeros dos elementos son una permutación de {4,5} y los últimos tres son una permutación de {1,2,3}. El valor "SIEMPRE" para $t_{reparacion}$ significa que todas las soluciones semifactibles se reparan, es decir, $umbralR$ es $+\infty$ durante toda la ejecución (se ignoran las líneas 8–9 del Algoritmo 7); y el valor "FIJO" significa que $umbralR$ se actualiza por única vez y conserva ese valor por el resto de la ejecución (se ignora la línea 10 del Algoritmo 7).

4.3. Resultados

Una vez configurados los parámetros, se procede a evaluar los algoritmos desarrollados para las dos etapas del abordaje secuencial del problema sobre el resto de las

Parámetro	Configuraciones	Mejor valor
α	0.05, 0.1, 0.2, 0.3	**0.1**
descenso	"PRIMER", "MEJOR"	**"PRIMER"**
ordenMov	[p{4,5}, p{1,2,3}]	[4,5,1,2,3]
	[4,5,1,2,3], [5,1,2,3], [4,1,2,3], [4,5,2,3], [4,5,1,3], [4,5,1,2], [4,5], [1,2,3]	**[4,5,1,2,3]**
$t_{reparacion}$	"SIEMPRE", "FIJO", 0.01, 0.05	**0.01**
descensoR	"PRIMER", "MEJOR"	**"PRIMER"**
ordenMovR	[6, p{4,5}, p{1,2,3}], [p{4,5}, 6, p{1,2,3}], [p{4,5}, p{1,2,3}, 6]	[6,5,4,1,2,3]
	[6,5,4,1,2,3], [5,4,1,2,3], [6,4,1,2,3], [6,5,1,2,3] [6,5,4,2,3], [6,5,4,1,3], [6,5,4,1,2]	[6,4,1,2,3], [6,5,4,1,3]
	[6,4,1,2,3], [6,5,4,1,3], [4,1,2,3], [6,1,2,3], [6,4,2,3], [6,4,1,3], [6,4,1,2]	[6,4,1,3]
	[6,4,1,3], [4,1,3], [6,1,3], [6,4,3], [6,4,1]	**[6,4,1,3]**
maxPerturb	1, 10, 25, 50, 75, 100, 250, 500, 1000	**100**

Cuadro 4.2.2.: Ajuste de parámetros de GRASP.

instancias. A continuación, se describen las experiencias computacionales llevadas a cabo y se analizan los resultados obtenidos.

4.3.1. Etapa I: LNS vs. CPLEX

La primera de las experiencias computacionales se ocupa de estudiar el comportamiento del algoritmo LNS para resolver la primera etapa del problema y de comparar los resultados obtenidos con los conseguidos por un solver de PLE al optimizar la formulación E1. Dado que LNS tiene ciertos componentes no deterministas, se hace un promedio de 5 corridas por instancia, limitando el tiempo de ejecución a 60 s por repetición (tiempo suficiente para hacer aproximadamente 20k iteraciones para las instancias del `Set1.1`). Respecto al solver, se usa CPLEX [IBM, 2021] en su versión 20.1.0, con todos sus parámetros por defecto, salvo un único thread y un límite de tiempo de 3600 s por instancia. En ambos casos, se optimiza la función objetivo normalizada descripta por la Ecuación 3.2.

En los Cuadros 4.3.1 y 4.3.2, se muestran los resultados obtenidos sobre las instancias del `Set1.1`. Para cada una de las instancias, los algoritmos se ejecutan variando λ en $\{0; 0{,}25; 0{,}5; 0{,}75; 1\}$. La columna "$cota_{inf}$" reporta la cota inferior del solver al terminar la ejecución, "f^*" es el valor objetivo de la mejor solución encontrada dentro del límite de tiempo, "$\%_{gap}$" es el porcentaje de gap relativo entre la cota inferior y el valor objetivo de la mejor solución (en el caso de LNS, se hace un promedio de todos los gaps relativos, utilizando también la cota inferior aportada por el solver) y "$t(s)$" es el tiempo de ejecución medido en segundos.

Se puede observar que CPLEX tiene claras dificultades en resolver las instancias consideradas, las cuales tienen en promedio 12k variables y 12k restricciones. De hecho, únicamente es capaz de probar optimalidad en 8 instancias y finaliza con un valor de gap positivo en 11, mientras que en las 26 restantes no encuentra ninguna solución factible dentro del tiempo límite. La resolución exacta de las instancias parece ser más difícil cuanto menor es el número de camiones, cuanto mayor es el horizonte de planificación y cuanto mayor es el valor de λ.

Por el contrario, LNS encuentra rápidamente soluciones para todas las instancias y los gaps reportados son bastante similares a los del solver, incluso llega a encontrar algunas soluciones óptimas. Para aquellas instancias en donde el solver encuentra

H	$\|V\|$	$\|C\|$	λ	CPLEX				LNS	
				$cota_{inf}$	f^*	$\%_{gap}$	$t(s)$	f^*	$\%_{gap}$
7	14	49	0	3,2211	–	–	3600	3,2620	1,3
			0,25	2,4652	–	–	3600	2,6560	7,7
			0,5	1,7097	–	–	3600	1,9509	14,1
			0,75	0,9275	–	–	3600	1,1508	24,1
			1	0,0768	–	–	3600	0,2699	251,4
7	21	53	0	1,8438	1,8438	0,0	512	1,8488	0,3
			0,25	1,4250	1,4593	2,4	3600	1,4896	4,5
			0,5	0,9956	1,0499	5,2	3600	1,0523	5,7
			0,75	0,5513	0,5768	4,4	3600	0,5766	4,6
			1	0,0579	–	–	3600	0,0579	0,0
7	28	56	0	2,0341	2,0341	0,0	11	2,0341	0,0
			0,25	1,5546	1,5580	0,2	3600	1,5703	1,0
			0,5	1,0665	1,0699	0,3	3600	1,0755	0,8
			0,75	0,5705	0,5720	0,3	3600	0,5774	1,2
			1	0,0620	–	–	3600	0,0620	0,0
14	10	35	0	2,0447	–	–	3600	2,0833	1,9
			0,25	1,6021	–	–	3600	1,7144	7,0
			0,5	1,1310	–	–	3600	1,2319	8,9
			0,75	0,6231	–	–	3600	0,6955	11,6
			1	0,0782	–	–	3600	0,0947	21,1
14	15	38	0	1,5569	1,5569	0,0	1244	1,5757	1,2
			0,25	1,2096	1,2337	1,9	3600	1,2607	4,2
			0,5	0,8548	–	–	3600	0,8925	4,4
			0,75	0,4832	–	–	3600	0,4960	2,7
			1	0,0887	–	–	3600	0,0887	0,0
14	20	40	0	1,3646	1,3646	0,0	8	1,3665	0,1
			0,25	1,0613	1,0616	0,02	3600	1,0743	1,2
			0,5	0,7383	0,7383	0,0	2176	0,7459	1,0
			0,75	0,3986	0,3986	0,0	283	0,4024	0,9
			1	0,0437	0,0437	0,0	906	0,0437	0,0

Cuadro 4.3.1.: Comparación entre CPLEX y LNS sobre las instancias del `Set1.1` (parte I).

H	$\|V\|$	$\|C\|$	λ	CPLEX				LNS	
				$cota_{inf}$	f^*	$\%_{gap}$	$t(s)$	f^*	$\%_{gap}$
28	6	21	0	1,8500	–	–	3600	1,8686	1,0
			0,25	1,4226	–	–	3600	1,5223	7,0
			0,5	0,9881	–	–	3600	1,0752	8,8
			0,75	0,5275	–	–	3600	0,5877	11,4
			1	0,0297	–	–	3600	0,0334	12,5
28	9	23	0	1,5684	–	–	3600	1,5822	0,9
			0,25	1,2063	–	–	3600	1,2542	4,0
			0,5	0,8292	–	–	3600	0,8653	4,3
			0,75	0,4394	–	–	3600	0,4599	4,6
			1	0,0250	–	–	3600	0,0250	0,0
28	12	24	0	1,6507	1,6507	0,0	1962	1,6552	0,3
			0,25	1,2582	1,2727	1,1	3600	1,2968	3,1
			0,5	0,8622	0,8782	1,8	3600	0,8943	3,7
			0,75	0,4547	0,4552	0,1	3600	0,4655	2,4
			1	0,0231	–	–	3600	0,0231	0,0

Cuadro 4.3.2.: Comparación entre CPLEX y LNS sobre las instancias del `Set1.1` (parte II).

al menos una solución factible, el gap reportado por el solver varía entre un 0 % y un 5,2 % mientras que el de LNS varía entre un 0 % y un 5,7 %. También es llamativo que LNS reporta un gap particularmente alto, mayor a 10 %, en algunas de las instancias. Todas ellas tienen en común que el número de camiones es bajo, que el valor de λ es alto y que el solver no encuentra soluciones factibles. Esto puede deberse a diversos motivos. Por ejemplo, la cota inferior devuelta por el solver podría estar en realidad muy alejada del valor óptimo. Otra posibilidad es que los parámetros del algoritmo podrían requerir valores diferentes para estas instancias particulares.

4.3.2. Etapa II: GRASP vs. GRASP-R vs. GRASP×ILS

En la próxima experiencia computacional se compara el desempeño de GRASP, GRASP-R y GRASP×ILS para la resolución de la segunda etapa del problema. En este caso, no es posible comparar los resultados obtenidos con los conseguidos por un solver de PLE sobre la formulación E2, debido a que la cantidad de variables y restricciones involucradas es intratable.

Los algoritmos se evalúan sobre las instancias provenientes del `Set1.1`. Específicamente, para cada una de ellas y para cada $\lambda \in \{0; 0{,}25; 0{,}5; 0{,}75; 1\}$, se aplica el algoritmo LNS para resolver la primera etapa del problema, con $nIters = 50000$ y el ajuste de parámetros sugerido por irace. Esto da como resultado un total de 45 instancias para iniciar los algoritmos basados en GRASP, en los cuales se usa $t_{lim} = 3600$ s y el ajuste de parámetros sugerido por irace. En consecuencia, para este experimento se destina en total un tiempo de CPU de a lo sumo 135 h.

Los resultados se presentan en el Cuadro 4.3.3a. Las primeras tres columnas describen los parámetros de la instancia (H, $|V|$ y $|C|$), y cada fila resume los resultados de las 5 instancias con esos mismos parámetros, pero con diferente valor de λ. La columna "$|T|$" es el promedio del número de tareas y las siguientes columnas reportan los resultados obtenidos por las metaheurísticas. La columna "sol" es el número de instancias donde al menos una solución factible fue encontrada, "f^*_{TAXI}" es el promedio del mejor valor objetivo entre las instancias resueltas, "$it(k)$" es el promedio del número de iteraciones totales (en miles), "$t(s)$" es el promedio de tiempo en que se encontró la mejor solución entre las instancias resueltas (en segundos) y "$\%_f$" es el porcentaje promedio de iteraciones fallidas entre las instancias resueltas (i.e. iteraciones de GRASP en donde no se encontró una solución factible). El valor más alto de "sol" entre las tres metaheurísticas aparece en negrita, y en caso de empate, el valor más bajo de "f^*_{TAXI}" también aparece en negrita.

El Cuadro 4.3.3b muestra los mismos resultados, esta vez proyectados por H y λ, i.e. cada fila resume los resultados de las 3 instancias con iguales valores de H y λ, pero con diferentes valores de $|V|$ y $|C|$. El valor "> 99" en la columna "$\%_f$" indica que el porcentaje promedio de iteraciones fallidas es en realidad más cercano a 100 que a 99.

El número de instancias resueltas por GRASP es 37 de 45, mientras que 5 instancias adicionales son resueltas por GRASP-R y GRASP×ILS. Ante igual número de instancias resueltas, GRASP×ILS reporta casi siempre mejor valor objetivo. En algunos casos, GRASP reporta menor valor objetivo promedio (por ejemplo para $H = 7$, $|V| = 28$ y $|C| = 56$), sin embargo, esto se debe al menor número de instancias resueltas (y en las instancias no resueltas, las demás metaheurísticas encuentran soluciones con alto valor objetivo). Al examinar los resultados de cada instancia individual, GRASP encuentra mejor solución en únicamente 2 de las 37

H	$\lvert V\rvert$	$\lvert C\rvert$	$\lvert T\rvert$	GRASP					GRASP-R					GRASP×ILS				
				sol	f^*_{TAXI}	*it*(*k*)	*t*(*s*)	$\%_f$	*sol*	f^*_{TAXI}	*it*(*k*)	*t*(*s*)	$\%_f$	*sol*	f^*_{TAXI}	*it*(*k*)	*t*(*s*)	$\%_f$
7	14	49	462	**5**	4.8	37.2	1339	98	**5**	**2.8**	9.8	698	73	**5**	4.4	2.9	1231	76
	21	53	448	**5**	13.2	18.6	456	64	**5**	11.2	12.2	621	45	**5**	**10.0**	2.1	538	44
	28	56	445	4	0.0	10.2	27	59	**5**	12.4	0.6	620	49	**5**	**10.4**	0.3	334	48
14	10	35	496	4	1.0	24.8	630	66	**5**	17.6	10.3	1070	61	**5**	**13.6**	1.0	785	60
	15	38	494	**5**	31.6	12.1	172	55	**5**	22.4	1.1	199	47	**5**	**20.4**	0.7	324	49
	20	40	480	**4**	6.0	40.7	590	81	**4**	4.0	10.5	650	58	**4**	**1.0**	1.5	724	58
28	6	21	497	3	21.3	100.3	798	97	**4**	41.5	11.1	1333	75	**4**	**39.5**	2.0	754	76
	9	23	497	4	71.0	72.7	3109	82	**5**	96.0	17.2	2077	74	**5**	**75.6**	3.0	2024	73
	12	24	495	3	24.7	100.2	1520	99	**4**	44.5	15.2	1544	75	**4**	**29.5**	1.8	1924	74

(a) Proyección por H, $\lvert V\rvert$, y $\lvert C\rvert$.

H	λ	$\lvert T\rvert$	GRASP					GRASP-R					GRASP×ILS				
			sol	f^*_{TAXI}	*it*(*k*)	*t*(*s*)	$\%_f$	*sol*	f^*_{TAXI}	*it*(*k*)	*t*(*k*)	$\%_f$	*sol*	f^*_{TAXI}	*it*(*k*)	*t*(*s*)	$\%_f$
7	0	425	**3**	**0.0**	5.3	286	37	**3**	**0.0**	0.02	2	20	**3**	**0.0**	0.03	7	25
	0.25	429	**3**	1.3	22.1	230	85	**3**	**0.0**	1.0	121	69	**3**	**0.0**	0.3	230	72
	0.5	432	**3**	**0.0**	1.3	68	82	**3**	**0.0**	0.1	13	41	**3**	**0.0**	0.4	290	34
	0.75	445	**3**	**4.0**	28.9	982	78	**3**	**4.0**	20.3	667	55	**3**	5.3	4.8	404	54
	1	528	2	37.0	52.4	2192	>99	**3**	40.0	16.2	2428	94	**3**	**36.0**	3.2	2575	95
14	0	448	**3**	5.3	40.8	914	79	**3**	2.7	18.1	1113	55	**3**	**0.0**	0.5	403	59
	0.25	451	**3**	4.0	27.5	398	78	**3**	4.0	10.5	440	59	**3**	**1.3**	1.5	444	61
	0.5	465	**3**	**0.0**	0.4	36	43	**3**	**0.0**	0.4	46	36	**3**	**0.0**	0.1	71	32
	0.75	476	**3**	**0.0**	3.9	382	54	**3**	**0.0**	3.9	543	40	**3**	**0.0**	1.1	631	40
	1	611	1	158.0	56.7	542	>99	**2**	98.0	3.6	1257	>99	**2**	**85.0**	2.0	1894	>99
28	0	458	1	188.0	121.1	3118	>99	**3**	134.7	5.9	1642	99	**3**	**103.3**	3.0	2170	99
	0.25	463	**3**	**15.3**	86.7	1320	86	**3**	18.0	24.4	3022	63	**3**	21.3	3.3	2127	63
	0.5	469	**3**	18.7	61.2	2484	89	**3**	15.3	18.2	925	60	**3**	**13.3**	1.7	733	60
	0.75	485	**3**	44.0	99.3	1620	98	**3**	44.0	22.3	1170	68	**3**	**33.3**	2.5	715	68
	1	605	0	–	–	–	–	**1**	188.0	1.7	1615	>99	**1**	**140.0**	1.1	3598	>99

(b) Proyección por H y λ.

Cuadro 4.3.3.: Desempeño de GRASP, GRASP-R y GRASP×ILS sobre `Set1.1`.

instancias resueltas.

El porcentaje de fallos de GRASP es de 76 % en promedio, reduciéndose a 61 % para las otras dos metaheurísticas. La gran cantidad de fallos evidencia la dificultad de construir soluciones factibles para este problema. El número de iteraciones de GRASP es de 46,3k en promedio, reduciéndose a 9,8k en GRASP-R, y a 1,7k en GRASP×ILS. En la mayoría de los casos, el tiempo en que se encuentra la mejor solución es considerablemente inferior a 3600 segundos, es decir, son encontradas mucho antes de finalizar la ejecución.

Las instancias más desafiantes resultan ser las de $\lambda = 1$ (i.e. priorizando la minimización de la demora en las entregas), evidenciado en el menor número de instancias resueltas, los mayores valores objetivo de las soluciones y el mayor porcentaje de fallos en comparación con otros valores de λ. Una explicación para este fenómeno puede ser el mayor número de tareas que presentan estas instancias, debido a que los camiones tienden a viajar mayores distancias en virtud de entregar sin demora los pedidos.

En algunos casos, se encuentran soluciones con garantía de optimalidad respecto a la segunda etapa, precisamente aquellas donde $f^*_{\text{TAXI}} = 0$ (no precisan taxis). Sin embargo, no es posible garantizar que sean globalmente óptimas al considerar los tres objetivos en simultáneo (por ejemplo, podría existir una solución que tampoco requiera taxis, pero en la cual los camiones recorran una distancia menor). Este tipo de soluciones tiene un interés particular porque también son soluciones de una posible variante del problema que prohíba viajes en taxi.

Finalmente, se vuelve a ejecutar GRASP×ILS con las 3 instancias no resueltas, usando un tiempo de ejecución 10 veces mayor ($t_{lim} = 10$ horas). Para la instancia con $H = 28$ y $|V| = 6$, se encontró una solución factible con valor objetivo igual a 152. En las otras dos instancias, ninguna solución fue encontrada, señalando que la dificultad estaría en las instancias por sí mismas y no en la escasez de tiempo de ejecución.

H	$\|V\|$	$\|C\|$	$\|T\|$	1 o 2 conductores						1 conductor				
				sol	f^*_{TAXI}	$it(k)$	t_s	$\%_f$	$\%_2$	*sol*	f^*_{TAXI}	$it(k)$	t_s	$\%_f$
7	14	49	462	**5**	**4.4**	2.9	1231	76	3.3	**5**	28.4	4.5	1316	78
	21	53	448	**5**	**10.0**	2.1	538	44	2.6	**5**	34.8	8.3	109	51
	28	56	445	**5**	10.4	0.3	334	48	3.4	4	3.0	2.4	539	44
14	10	35	496	**5**	**13.6**	1.0	785	60	3.2	**5**	50.4	4.2	1726	63
	15	38	494	**5**	**20.4**	0.7	324	49	3.0	**5**	45.6	6.3	946	52
	20	40	480	**4**	**1.0**	1.5	724	58	4.2	**4**	28.0	5.6	1500	62
28	6	21	497	**4**	**39.5**	2.0	754	76	5.1	**4**	75.5	4.8	1542	75
	9	23	497	**5**	**75.6**	3.0	2024	73	4.9	**5**	98.4	5.5	1623	74
	12	24	495	**4**	**29.5**	1.8	1924	74	5.9	**4**	93.5	4.7	2372	74

Cuadro 4.3.4.: Desempeño de GRASP×ILS al variar el tamaño de las tripulaciones sobre `Set1.1`.

4.3.3. Conductores: 1 vs. 2

La finalidad de esta experiencia computacional es estudiar el impacto que tiene una de las principales características del problema en estudio sobre la calidad de las soluciones encontradas por la metaheurística. Esto se refiere a la opción de que viajen hasta dos conductores a la vez en los camiones, siendo posible, en teoría, reducir la dependencia a los viajes en taxi.

Se ejecuta GRASP×ILS de forma análoga al experimento anterior, salvo que esta vez se asigna $ordenMov = [1, 2, 3]$ y $ordenMovR = [6, 1, 3]$, es decir, se apagan Mov. 4 y 5. Al apagar estos movimientos, los cuales son los únicos capaces de modificar el número de tripulantes, las soluciones semifactibles encontradas por la búsqueda local preservan la propiedad de que cada tarea está asignada a un único conductor, ya que así son construidas inicialmente por la heurística constructiva.

Los resultados se presentan en el Cuadro 4.3.4. Las columnas 5–10, repiten los resultados obtenidos por GRASP×ILS en la experiencia anterior, es decir, con tripulaciones de 1 o 2 conductores. Se agrega una nueva columna "$\%_2$" que indica el porcentaje promedio de tareas que tienen 2 conductores asignados en la mejor solución encontrada. Las columnas 11–15, reportan los resultados obtenidos al apagar Mov. 4 y 5.

Cuando no se permiten conductores adicionales, la metaheurística encuentra solu-

ciones factibles en 41 de 45 instancias (una menos que antes). Lo más llamativo es el marcado deterioro que se observa en los valores objetivos (salvo para $H = 7$, $|V| = 28$ y $|C| = 56$, donde se reporta un promedio más bajo pero es debido a la instancia no resuelta en ese grupo). Esto sugiere que la frecuencia con que los conductores necesitan tomar taxis se incrementa cuando las tripulaciones se restringen. Otra diferencia notable es la mayor cantidad de iteraciones, aunque por el contrario el porcentaje de fallos se mantiene parejo. El porcentaje de tareas que tienen asignado a dos conductores varía del 2.6 % al 5.9 %, tendiendo a crecer cuanto mayor es el horizonte de planificación.

4.3.4. Límites y otras regulaciones laborales

Por último, es interesante estudiar el comportamiento de la metaheurística sobre grandes instancias para identificar sus límites, así como también investigar su aplicabilidad al incluir otras legislaciones laborales, a pesar de no ser requeridas originalmente por la empresa donde surge el problema en estudio. En este sentido, es usual encontrar en la literatura restricciones que limiten el número de horas de trabajo acumuladas en la semana y restricciones que impongan duraciones mínimas sobre los descansos, véase por ejemplo las regulaciones que rigen en la UE, las cuales se pueden encontrar en [Drexl et al., 2013]. En este caso, se proponen las siguientes legislaciones laborales:

L_1. Cada conductor debe descansar al menos 12 horas en todo intervalo de 24 horas y debe tener al menos un día de franco en todo intervalo de 7 días (legislación original).

L_2. Cada conductor debe trabajar a lo sumo 60 horas en todo intervalo de 7 días.

L_3: Cada conductor debe descansar ininterrumpidamente por al menos 11 horas entre todo par de períodos de trabajo consecutivos.

La incorporación de las nuevas reglas a los algoritmos ya desarrollados requiere de algunas modificaciones. Una de las decisiones más importantes involucra adaptar la definición de solución semifactible. Para minimizar el número de cambios en el código, se propone considerar a toda solución semifactible que viole L_2 o L_3 directamente como infactible. Por supuesto, otros enfoques también son válidos.

				L_1					L_1+L_2					L_1+L_3				
$\|P\|$	H	$\|V\|$	$\|C\|$	*sol*	f^*_{TAXI}	*it*	$t(s)$	$\%_f$	*sol*	f^*_{TAXI}	*it*	$t(s)$	$\%_f$	*sol*	f^*_{TAXI}	*it*	$t(s)$	$\%_f$
1000	7	160	360	5	8,0	62	1100	61	5	8,0	76	656	62	4	39,5	451	1297	91
			400	5	0,0	29	345	40	5	0,0	29	197	40	4	11,5	283	1985	69
	14	120	270	3	1,3	131	2430	84	4	2,5	168	1215	87	0	–	615	–	–
			300	4	0,0	74	1025	54	4	0,0	54	431	65	4	51,0	395	1615	98
	28	80	180	5	5,6	97	1244	66	4	0,0	102	766	59	4	51,5	744	2043	83
			200	5	0,0	30	339	28	5	0,0	61	395	23	5	18,4	232	1310	69
3000	7	520	1040	3	6,7	17	1958	76	3	6,7	22	1149	78	2	21,0	131	658	95
			1170	5	0,0	2	180	15	5	0,0	2	110	15	3	9,3	73	911	89
	14	410	820	4	1,5	2	843	59	5	5,6	16	769	42	1	4,0	98	164	95
			922	5	0,0	2	113	15	5	0,0	2	73	15	4	2,0	28	749	56
	28	300	600	5	0,0	2	115	13	5	0,0	4	126	14	5	8,8	23	185	33
			675	5	0,0	1	43	0	5	0,0	1	29	0	5	0,0	8	376	15

Cuadro 4.3.5.: Desempeño de GRASP×ILS con diferentes regulaciones de descanso sobre `Set1.2`.

Por ejemplo, si la construcción de soluciones semifactibles se vuelve demasiado difícil, se podría permitir violar L_2 y/o L_3 durante la construcción inicial y luego aplicar una heurística de reparación.

En este experimento computacional se utilizan las instancias del `Set1.2`. Para cada una de ellas y para cada $\lambda \in \{0; 0{,}25; 0{,}5; 0{,}75; 1\}$, se aplica el algoritmo LNS para resolver la primera etapa, con un tiempo límite de 15 minutos por ejecución y $\alpha = 0{,}03$ (en cada iteración se destruyen y reconstruyen a lo sumo el 3 % de los pedidos). En cada una de las 60 instancias resultantes, se ejecuta GRASP×ILS con la misma configuración que en el experimento anterior ($t_{lim} = 3600$ s), pero variando las legislaciones. En el Cuadro 4.3.5, se resumen los resultados obtenidos. Las columnas 5–9 usan la legislación original L_1, las columnas 10–14 aplican L_1+L_2 (se imponen L_1 y L_2 simultáneamente) y las columnas 15–19 consideran L_1+L_3 (se imponen L_1 y L_3 simultáneamente). En este caso, el promedio del número de iteraciones "*it*" se mide en unidades (y no en miles).

En las instancias con $|P| = 1000$, se puede apreciar que el comportamiento es bastante similar al incorporar L_2. De hecho, se mantiene el mismo número de instancias resueltas, 27 de 30, y se encuentran soluciones factibles de igual costo para aquellas con $H = 7$ y con $H \in \{14, 28\}$ y $|C| = 2,5|V|$. Por el contrario, el comportamiento es muy diferente al incorporar L_3. En particular, es posible

resolver únicamente 21 de 30 instancias y no se encuentra ninguna solución factible en aquellas con $H = 14$ y $|C| = 2,25|V|$. Además, se observa que la búsqueda de soluciones factibles se complejiza, dado el gran aumento en el promedio del número de iteraciones totales y del porcentaje de iteraciones fallidas (recordar que la construcción descarta soluciones semifactibles que violen L_3). Esto es más evidente en instancias donde el número de conductores es más ajustado.

En las instancias con $|P| = 3000$, se aprecian los mismos comportamientos que ya fueron destacados. Por otro lado, es evidente que las iteraciones son ahora más pesadas, dado que es posible hacer muchas menos iteraciones dentro del tiempo límite. Además, es llamativa la disminución observada en el promedio de iteraciones fallidas, lo que indicaría que estas instancias son en realidad *más fáciles* de resolver en comparación con aquellas con $|P| = 1000$. Esto último se puede atribuir a la proporción de pedidos, camiones y conductores que estas instancias poseen.

Surge entonces la pregunta de si la imposibilidad de encontrar soluciones factibles se debe a debilidades en la metaheurística o a falta de tiempo de ejecución. Para intentar dar una respuesta, se vuelve a ejecutar GRASP×ILS con la regulación L_1+L_3 sobre la instancia con $|P| = 3000$, $H = 7$, $|V| = 520$, $|C| = 1170$ y $\lambda = 0{,}5$ (para la cual no fue posible encontrar solución factible), aumentando el tiempo límite a $t_{lim} = 10$ horas. Es posible encontrar ahora una solución factible con un costo igual a 22, encontrada en la iteración número 335 de un total de 745. Por lo tanto, es probable que el límite de tiempo de 1 hora sea insuficiente cuando el número de pedidos es grande y la legislación L_3 está presente.

5. Conclusiones

En este trabajo, se aborda un complejo problema de logística en el transporte terrestre, el cual combina el ruteo de una flota de camiones con la planificación de sus tripulaciones. El objetivo es cumplir con un conjunto de pedidos de recolección y entrega de mercadería con ventanas de tiempo dentro de un horizonte de planificación en el menor costo posible, el cual depende de la distancia recorrida por los camiones de la compañía y por un servicio de transporte externo y de la demora en las entregas. Dado que el transporte involucra largas distancias y el horizonte múltiples días, se vuelve imprescindible el relevo de los trabajadores para optimizar el uso de los camiones.

Durante esta primera parte de la tesis, se trabaja con una descomposición secuencial del problema en dos subproblemas más simples. Mientras que el primero involucra la construcción de rutas factibles para los camiones, el segundo se ocupa de asignar a los conductores sobre las rutas ya construidas, aunque permitiendo modificar algunas de las decisiones tomadas previamente. La gran ventaja de la descomposición secuencial radica en desacoplar la optimización de los camiones con la de los conductores, lo que permite abordar instancias mucho más grandes.

Se propone un algoritmo basado en la metaheurística LNS para resolver el primer subproblema. Las experiencias computacionales dan cuenta de su efectividad, visto la calidad de las soluciones encontradas. A pesar de que todavía existe margen de mejora, hay que tener en cuenta que el objetivo de esta primera etapa no es simplemente encontrar soluciones óptimas, sino encontrar variedad de soluciones factibles de buena calidad para aumentar las oportunidades de éxito en la segunda etapa, y sobre todo que las soluciones encontradas sean globalmente buenas. Por otro lado, tampoco debería preocupar el comportamiento observado cuando se prioriza las entregas sin demora sobre los costos de viaje, ya que tales situaciones son poco reales en la práctica.

Se proponen tres algoritmos para resolver el segundo subproblema. El primero de ellos se basa en la metaheurística GRASP, el segundo incorpora una heurística de reparación y el tercero una heurística de perturbación, convirtiéndolo en una metaheurística híbrida GRASP×ILS. Las experiencias computacionales confirman que la posibilidad de violar los descansos en los intervalos de 24 horas durante la construcción de las soluciones iniciales, que posteriormente son reparadas heurísticamente, es clave para mejorar la factibilidad del algoritmo. Además, la opción de perturbar el horario asignado para algunas de las tareas antes de reiniciar la búsqueda, es fundamental para mejorar la calidad de las soluciones encontradas. Por lo tanto, y a pesar de que cada iteración se vuelve computacionalmente más pesada, el desempeño de GRASP×ILS es superior al de las otras metaheurísticas estudiadas, validando el enfoque híbrido propuesto.

Con respecto a la principal característica de este problema, se puede decir que, en larga distancia, la posibilidad de llevar a un conductor adicional conduce a notables mejoras en los costos, los cuales son en promedio 2,25 veces menores. En otras palabras, los conductores necesitan con mayor frecuencia costosos transportes externos cuando las tripulaciones se limitan a un único conductor.

Por otra parte, el algoritmo basado en GRASP×ILS es capaz de dar soluciones a instancias grandes, con hasta 3000 pedidos, provisto de un tiempo de ejecución razonable. También es robusto, en el sentido de que puede ser adaptado fácilmente para contemplar otras variantes en la legislación laboral que rige los descansos de los conductores, aunque los resultados obtenidos pueden ser disímiles según la legislación. Por ejemplo, en legislaciones que incorporen duraciones mínimas de descanso consecutivo, la construcción de soluciones iniciales puede resultar seriamente comprometida. En estos casos, puede ser interesante permitir que tales restricciones sean violadas durante la construcción inicial y luego aplicar una heurística de reparación, de la misma forma que se hizo en la tesis con los descansos en los intervalos de 24 horas.

Parte II.

Enfoque simultáneo

6. Modelado con Programación Lineal Entera en una etapa

El enfoque secuencial es altamente efectivo para dar solución a grandes instancias del problema, pero tiene algunas limitaciones. Por un lado, no garantiza la obtención de soluciones, dado que las rutas de camiones construidas en la primera etapa podrían no admitir ninguna asignación factible de tripulantes en la segunda, a pesar de que la instancia sea factible para el problema original. Incluso siendo factibles las instancias para cada uno de los subproblemas, la resolución exacta de cada una de ellas tampoco garantiza la obtención de una solución globalmente óptima. Por ejemplo, el mejor plan de rutas para los camiones podría conducir a una planificación de tripulaciones demasiado costosa, mientras que uno relativamente bueno podría dar lugar a planificaciones mucho más económicas, conduciendo a mejores costos globales cuando los objetivos se combinan en una única función lineal de suma ponderada. La resolución exacta de los subproblemas tampoco garantiza optimalidad en el sentido de Pareto, pues es posible que existan diferentes planes de rutas para los camiones que sean Pareto optimales y con mismo costo de viaje y de demora, pero que la planificación de tripulaciones tenga un valor óptimo distinto en cada uno de ellos. Por otro lado, el abordaje secuencial asume implícitamente un orden de prioridad entre los objetivos. Por ejemplo, sería muy difícil encontrar soluciones que prioricen el costo de la planificación de las tripulaciones por sobre los costos asociados al ruteo de los camiones.

La única forma de sortear estos inconvenientes es por medio de un enfoque íntegramente simultáneo, es decir, donde el ruteo de los vehículos y la planificación de las tripulaciones se realicen en una misma y única etapa. Este cambio de abordaje es investigado durante la segunda parte de la tesis y plantea nuevos desafíos, pues aparecen complejas restricciones de sincronización en tiempo y espacio entre

			Formulaciones de PLE					
			LT-LT	LT-LTX	LTC-LT	LTC-LTX	LTP-LT	LTP-LTX
Camiones	Digrafo	G_{LT}	✓	✓				
		G_{LTC}			✓	✓		
		G_{LTP}					✓	✓
	Variables X	Binarias	✓	✓	✓	✓		
		Enteras					✓	✓
	Restricciones sobre $E^{\text{R}} \cup E^{\text{E}}$	Emparejamiento	✓	✓	✓	✓		
		Precedencia	✓	✓	✓	✓		
		Capacidad	✓	✓				
Conductores	Digrafo	G_{LT}	✓		✓		✓	
		G_{LTX}		✓		✓		✓
	Variables	Y sobre E^{TAXI}		✓		✓		✓
		Z sobre E^{VIAJ}	✓		✓		✓	
	Restricciones sobre E^{VIAJ}	Sincronización simple	✓		✓		✓	
		Sincronización doble		✓		✓		✓
		Activación de Z	✓		✓		✓	

Cuadro 6.0.1.: Resumen de las formulaciones de PLE propuestas.

los camiones y los conductores. Particularmente en este capítulo, se proponen seis formulaciones de PLE diferentes para modelar el problema en una única etapa. Cada una de ellas recurre a dos digrafos auxiliares, uno para representar las rutas de los camiones y el otro para las rutas de los conductores, donde los nodos codifican localidades e instantes de tiempo y los arcos codifican desplazamientos. Los digrafos permiten representar rutas como caminos dirigidos, sin embargo es necesario incorporar a los modelos restricciones adicionales para prohibir ciertos caminos dirigidos que no representan rutas válidas. Estas restricciones externas inspiran la definición de digrafos más complejos que las codifican en su estructura, derivando en formulaciones de PLE con menos familias de restricciones. Si bien en un principio el modelado de las rutas se presenta de manera separada para los camiones y los conductores, posteriormente se introducen las restricciones necesarias para sincronizarlas.

En el Cuadro 6.0.1 se presenta a modo de resumen una comparación entre las formulaciones de PLE que se propondrán a lo largo del capítulo. Las diferentes categorías que se comparan se explicarán más adelante.

6.1. Digrafo LT – Localidad-Tiempo

Una forma natural de representar el desplazamiento espacio-temporal de objetos es mediante un digrafo que posea un nodo por cada posible ubicación e instante de tiempo, y un arco por cada posible transición. De esta forma, los caminos dirigidos en el digrafo describen por completo las acciones realizadas por el objeto a lo largo del tiempo. En el caso de estudio, los camiones y los conductores pueden hacer 4 tipos de transiciones: permanecer en una misma localidad, cargar o descargar un pedido, y viajar a otra localidad. Notar que en los primeros 3 casos, y a diferencia del último, el objeto permanece en la misma ubicación, pero se desplaza en el tiempo.

Con el objetivo de representar las rutas de camiones y de conductores como caminos dirigidos, se define un (multi)digrafo $G_{\text{LT}} = (N, E)$. El conjunto N contiene un nodo $n_{l,i}$ para cada $l \in L$ e $i \in \mathcal{I}$, y dos nodos especiales, la fuente n_f y el sumidero n_s. El multiconjunto E es la unión (disjunta)[1] de los siguientes conjuntos de arcos:

- E^{DESC} es el conjunto de arcos de descanso. Modelan que un camión está detenido o que un conductor está descansando por una unidad de tiempo. Contiene un arco de la forma $(n_{l,i}, n_{l,i+1})$ para cada $l \in L$ e $i \in \mathcal{I}$ con $i \leq I.H - 1$.

- E^{VIAJ} es el conjunto de arcos de viaje. Modelan que un camión o que un conductor se desplazan entre dos localidades. Contiene un arco de la forma $(n_{l_1,i}, n_{l_2,i+\Delta})$ para cada par $l_1, l_2 \in L$ de localidades adyacentes y distintas e $i \in \mathcal{I}$ con $i \leq I.H - \Delta$, donde $\Delta = \text{tiempoViajeCamion}(l_1, l_2)$.

- $E^{\text{R},p}$ es el conjunto de arcos de recolección asociados al pedido p. Modelan que un camión o un conductor realizan la carga del pedido. Contiene un arco de la forma $(n_{l_p^{\text{R}},i}, n_{l_p^{\text{R}},i+s_p^{\text{R}}})$ para cada $i \in \mathcal{I}_p^{\text{R}}$.

- $E^{\text{E},p}$ es el conjunto de arcos de entrega asociados al pedido p. Modelan que un camión o un conductor realizan la descarga del pedido. Contiene un arco de la forma $(n_{l_p^{\text{E}},i}, n_{l_p^{\text{E}},i+s_p^{\text{E}}})$ para cada $i \in \mathcal{I}_p^{\text{E}}$.

[1] En lugar de marcar cada arco del multiconjunto con un índice para identificar su procedencia, preferimos no sobrecargar la notación y mencionar de forma explícita el conjunto del cual proviene cuando haya ambigüedades.

- E^{FUEN} es el conjunto de arcos que salen de la fuente. Permiten modelar las rutas como caminos dirigidos que inician en el nodo n_f. Contiene un arco de la forma $(n_f, n_{l,0})$ para cada $l \in L$.

- E^{SUMI} es el conjunto de arcos que llegan al sumidero. Permiten modelar las rutas como caminos dirigidos que llegan al nodo n_s. Contiene un arco de la forma $(n_{l,I.H}, n_s)$ para cada $l \in L$.

A continuación, se presenta un ejemplo de esta construcción.

Ejemplo 6.1.1. En la Figura 6.1.1a, se muestra el digrafo G_{LT} que resulta del Ejemplo 1.2.1. La posición de los nodos en el eje vertical determina la localidad y en el eje horizontal, el instante de tiempo. Los arcos de descanso son paralelos al eje horizontal. Los arcos de viaje son oblicuos y no incidentes en la fuente ni el sumidero. Los arcos de carga y de descarga son curvos, y para identificarlos, se escribe sobre ellos una R. cuando sean de recolección y una E. cuando sean de entrega, seguidos por el nombre del pedido. Las Figuras 6.1.1b y 6.1.1c muestran los (n_f, n_s)-caminos dirigidos que se corresponden con las rutas de los camiones v_1 (naranja) y v_2 (cyan) y las rutas de los conductores c_1 (verde) y c_2 (magenta).

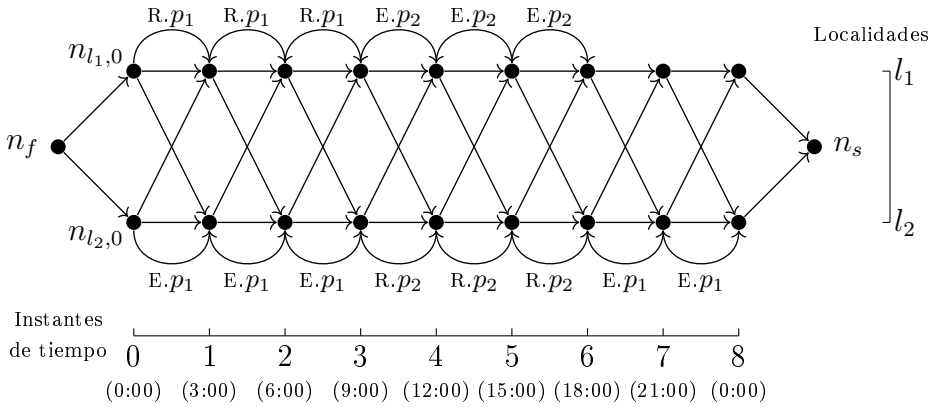
Por su construcción, G_{LT} es un digrafo acíclico, en donde toda ruta de camión y de conductor se corresponde con algún (n_f, n_s)-camino dirigido. Sin embargo la recíproca no es cierta, como se verá a continuación.

6.1.1. Rutas de conductores

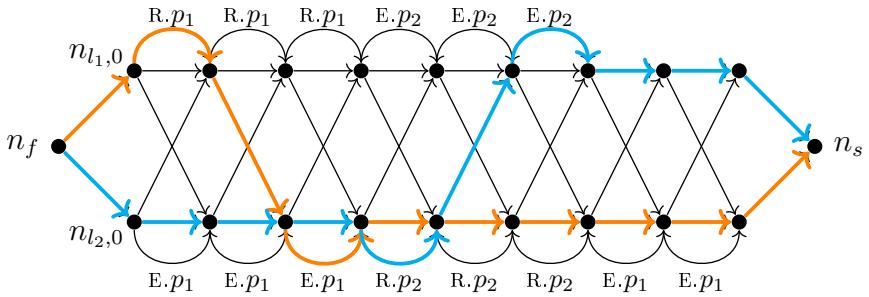
Las regulaciones en las horas de trabajo de los conductores imponen que los mismos deben:

- tener un descanso de al menos 12 horas en todo intervalo de 24 horas, y

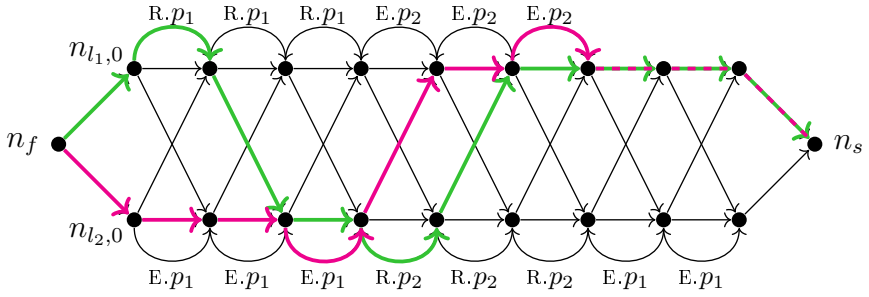
- tener al menos un día de franco en todo intervalo de 7 días.



(a) Digrafo G_{LT}.



(b) Rutas de camiones.



(c) Rutas de conductores.

Figura 6.1.1.: Ejemplos para el digrafo LT – Localidad-Tiempo.

En consecuencia, para que un (n_f, n_s)-camino dirigido se corresponda con una ruta de conductor deben verificarse adicionalmente dichas condiciones. En otras palabras, los (n_f, n_s)-caminos dirigidos no pueden pasar arbitrariamente por los arcos de viaje, de recolección y de entrega, sino que están obligados a pasar por arcos de descanso con cierta frecuencia. Los caminos que no cumplan estas restricciones son llamados *caminos prohibidos* y deben ser excluidos de las soluciones del modelo. En lo que sigue a continuación, se describe cómo modelar estos caminos usando PLE.

Dado un conductor c, se define el subconjunto $E^c \subset E$ de arcos por los cuales puede desplazarse c, esto es, todos los arcos salvo aquellos que salgan de la fuente y lleguen a un nodo que no coincida con la localidad inicial de c, es decir, $E^c = E \setminus \{(n_f, n_{l,0}) \in E^{\text{FUEN}} : l \neq l_c\}$. Ahora, se considera una variable $Y_{ce} \in \{0, 1\}$ para todo $c \in C$ y $e \in E^c$ tal que $Y_{ce} = 1$ si y solo si el conductor c se desplaza por el arco e. Estas variables están sujetas a las siguientes restricciones.

- *Restricciones de conservación de flujo*:

$$\sum_{e \in \Gamma^-(n) \cap E^c} Y_{ce} = \sum_{e \in \Gamma^+(n) \cap E^c} Y_{ce} \qquad \forall c \in C,\ n \in N \setminus \{n_f, n_s\}.$$

  Estas restricciones obligan a que si un conductor c pasa por un arco que llega a un nodo n, entonces también tiene que pasar por un arco que sale de n, y viceversa. Es decir, imponen que los arcos por los cuales pasa el conductor formen un camino dirigido en el digrafo.

- *Restricciones de descanso de 12 horas:*

$$\sum_{\substack{e \in E^{\text{DESC}}: \\ i \leq \text{ini}(e) < i+I}} Y_{ce} \geq \frac{I}{2} \qquad \forall c \in C,\ i \in \{0, \ldots, I.(H-1)\},$$

  donde ini es una función que, dado un arco de $E \setminus (E^{\text{FUEN}} \cup E^{\text{SUMI}})$, retorna el instante de tiempo en el que inicia el arco, es decir, si $e = (n_{l1,i1}, n_{l2,i2})$, entonces $\text{ini}(e) = i_1$.

  Estas restricciones obligan a que un conductor c pase por al menos $\frac{I}{2}$ arcos de descanso en cada intervalo de 24 horas. Como cada arco de descanso tiene una duración de un instante de tiempo y cada instante de tiempo equivale

a $\frac{24}{I}$ horas, resulta claro que el lado derecho de la restricción equivale a 12 horas.

- *Restricciones de descanso de 24 horas:*

  Para expresarlas, se requiere incorporar al modelo una variable binaria W_{ch} para cada $c \in C$ y $h \in \{0, \ldots, H-1\}$ tal que $W_{ch} = 1$ si y solo si el conductor c está de franco en el día h. Estas variables deben estar sujetas a las siguientes restricciones:

$$\sum_{h' \in \{h,\ldots,h+6\}} W_{ch'} \geq 1 \qquad \forall c \in C,\ h \in \{0, \ldots, H-7\},$$

$$\sum_{\substack{e \in E^{\text{DESC}}: \\ I.h \leq \text{ini}(e) < I.h+I}} Y_{ce} \geq I.W_{ch} \qquad \forall c \in C,\ h \in \{0, \ldots, H-1\}.$$

  Las primeras restricciones obligan a que un conductor c tenga al menos un día de franco en todo intervalo de 7 días. Mientras que las otras obligan a que un conductor c de franco en el día h pase por I arcos de descanso durante ese día, esto es, que descanse las 24 horas de ese día.

6.1.2. Rutas de camiones

El orden en que los camiones realizan las recolecciones y las entregas está sujeto a una serie de restricciones. Por un lado, los camiones tienen capacidad para cumplir un único pedido a la vez. Es decir, tras cargar una mercancía, el camión no puede cargar otra hasta que haya descargado la actual en su respectiva localidad de entrega. A su vez, todos los pedidos se deben cumplir exactamente una vez. Es decir, toda mercancía es cargada, una única vez, y es eventualmente descargada antes de que termine el horizonte de planificación. Aunque no fueron mencionadas en la descripción del problema, hay también ciertas reglas implícitas a cumplir. En particular, los camiones no pueden entregar una mercancía que no hayan recolectado previamente, o que ya hayan entregado.

En base a estas restricciones, se deduce que para que un (n_f, n_s)-camino dirigido se corresponda con una ruta de camión es necesario que sus arcos de recolección y

de entrega ocurran de la siguiente forma:

$$\text{R. } p_{i_1},\ \text{E. } p_{i_1},\ \ldots,\ \text{R. } p_{i_n},\ \text{E. } p_{i_n}$$

con $p_{i_1}, \ldots, p_{i_n} \in P$, distintos 2 a 2, y $n \in \mathbb{N}_0$. Es decir, el camino alterna arcos de recolección y de entrega asociados a un mismo pedido y no se repiten pedidos. En lo que sigue a continuación, se describe cómo modelar estos caminos usando PLE.

Dado un camión v, se define el subconjunto $E^v \subset E$ de arcos por los cuales puede desplazarse v, esto es, todos los arcos salvo aquellos que salgan de la fuente y lleguen a un nodo que no coincida con la localidad inicial de v, es decir, $E^v = E \setminus \{(n_f, n_{l,0}) \in E^{\text{FUEN}} : l \neq l_v\}$. Ahora, se considera una variable $X_{ve} \in \{0, 1\}$ para todo $v \in V$ y $e \in E^v$ tal que $X_{ve} = 1$ si y solo si el camión v se desplaza por el arco e. Estas variables están sujetas a las siguientes restricciones.

- *Restricciones de conservación de flujo*:

$$\sum_{e \in \Gamma^-(n) \cap E^v} X_{ve} = \sum_{e \in \Gamma^+(n) \cap E^v} X_{ve} \qquad \forall v \in V,\ n \in N \setminus \{n_f, n_s\}.$$

  Estas restricciones son similares a las de conservación de flujo para los conductores, pero esta vez expresadas para los camiones.

- *Restricciones de cumplimiento de pedidos:*

$$\sum_{e \in E^{\text{R},p}} \sum_{v \in V} X_{ve} = 1 \qquad \forall p \in P.$$
$$\sum_{e \in E^{\text{E},p}} \sum_{v \in V} X_{ve} = 1 \qquad \forall p \in P.$$

  Estas restricciones garantizan que todo pedido p es recolectado y entregado exactamente una vez por algún camión.

En el siguiente ejemplo, se muestran algunas soluciones que verifican estas restricciones pero que, por diferentes motivos, no se corresponden con rutas de camiones válidas.

Ejemplo 6.1.2. En Figura 6.1.2, se muestran algunas soluciones que no se corresponden con rutas de camiones válidas para la instancia del Ejemplo 1.2.1. Se utiliza la siguiente codificación: un arco e de color negro simboliza que las variables X_{v_1e} y X_{v_2e} están apagadas, si es de color naranja representa que la variable X_{v_1e} asume el valor 1 y la variable X_{v_2e} asume el valor 0, y lo opuesto si es de color cyan. En los (n_f, n_s)-caminos dirigidos de la Figura 6.1.2a, los arcos de recolección y entrega visitados por cada uno de ellos están asociados a pedidos diferentes. El (n_f, n_s)-camino dirigido de la Figura 6.1.2b visita arcos de recolección y de entrega asociados a un mismo pedido, pero en el orden incorrecto. El (n_f, n_s)-camino dirigido de la Figura 6.1.2c visita los arcos de recolección asociados a p_1 y a p_2 y luego sus arcos de entrega correspondientes, transportando así más de un pedido a la vez.

Para cortar estos caminos prohibidos, son necesarias las siguientes restricciones.

- *Restricciones de emparejamiento:*

$$\sum_{e \in E^{\mathrm{R},p}} X_{ve} = \sum_{e \in E^{\mathrm{E},p}} X_{ve} \qquad \forall v \in V,\ p \in P.$$

  Estas restricciones aseguran que un camión v recolecta un pedido p si y sólo si el mismo camión también lo entrega.

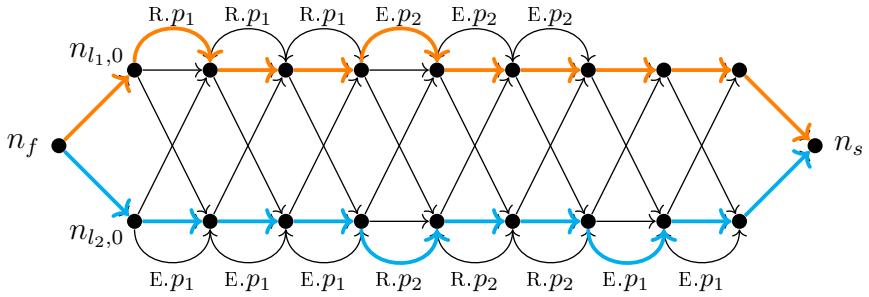
  Notar que estas restricciones hacen que una de las dos familias de restricciones de cumplimiento de pedidos quede implicada, con lo cual pueden ser omitidas del modelo.

- *Restricciones de precedencia:*

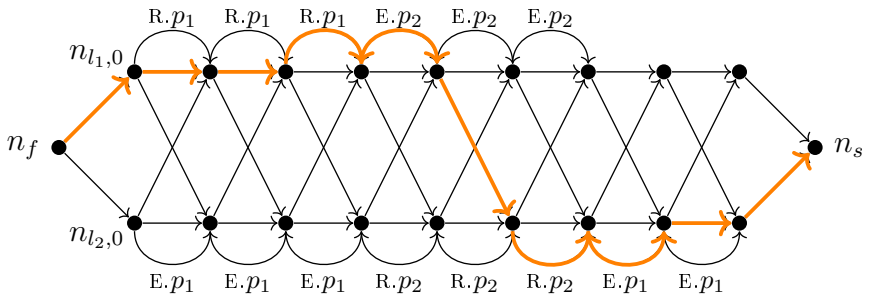
$$\sum_{\substack{e' \in E^{\mathrm{R},p}: \\ \mathrm{fin}(e') \leq \mathrm{ini}(e)}} \sum_{v \in V} X_{ve'} \geq \sum_{v \in V} X_{ve} \qquad \forall p \in P,\ e \in E^{\mathrm{E},p},$$

  donde fin es una función que, dado un arco de $E \setminus (E^{\mathrm{FUEN}} \cup E^{\mathrm{SUMI}})$, retorna el instante de tiempo en el que finaliza el arco, es decir, si $e = (n_{l1,i1}, n_{l2,i2})$, entonces $\mathrm{fin}(e) = i_2$.

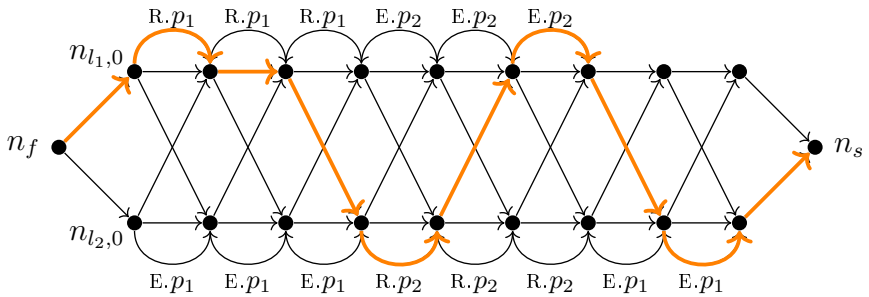
  Estas restricciones prohíben que un pedido pueda ser entregado antes de haber sido recolectado. Observar que, por la construcción de G_{LT}, no es a



(a) Se incumplen las restricciones de emparejamiento.



(b) Se incumplen las restricciones de precedencia.



(c) Se incumplen las restricciones de capacidad.

Figura 6.1.2.: Ejemplos de (n_f, n_s)-caminos dirigidos prohibidos en G_{LT}.

priori necesario incluir en ellas el tiempo de viaje entre las localidades de recolección y de entrega. Es decir, todo (n_f, n_s)-camino dirigido que pase por un arco e_1 de recolección y e_2 de entrega asociados a un mismo pedido p, ya verifican que $\text{fin}(e_1) + \text{tVC}(l_p^{\text{R}}, l_p^{\text{E}}) \leq \text{ini}(e_2)$. Sin embargo, en la sección 7.1.1 se volverá a discutir esta alternativa.

- *Restricciones de capacidad:*

$$\sum_{p \in P} \Big(\sum_{\substack{e \in E^{\text{R},p}: \\ \text{fin}(e) \leq i}} X_{ve} - \sum_{\substack{e \in E^{\text{E},p}: \\ \text{fin}(e) \leq i}} X_{ve} \Big) \leq 1 \qquad \forall v \in V,\ i \in \mathcal{I}^{\text{R}},$$

donde $\mathcal{I}^{\text{R}}$ es el conjunto de instantes de tiempo en los cuales pudo haberse recolectado alguna mercancía, es decir,

$$\mathcal{I}^{\text{R}} = \{\text{fin}(e) : p \in P, e \in E^{\text{R},p}\}.$$

Estas restricciones fuerzan a que, hasta el instante de tiempo i, la cantidad de mercancías cargadas por el camión v puede ser a lo sumo uno más que la cantidad de mercancías descargadas. En consecuencia, impiden que se carguen dos mercancías sin realizar una descarga intermedia.

A pesar de que las rutas de conductores y de camiones han sido presentadas de manera separada, es claro que al combinarlas se deben satisfacer restricciones de sincronización, las cuales se analizarán a continuación.

6.1.3. Sincronización de rutas

El problema requiere que los camiones tengan al menos uno y a lo sumo dos conductores cada vez que viajan, cargan y descargan mercancías. Recíprocamente, los conductores deben estar en algún camión cada vez que realicen una carga o una descarga, pero no así cuando hacen un viaje, ya que pueden desplazarse en taxi cuando no hay camiones disponibles. Estos requerimientos de sincronización son muy sencillos de incorporar en el modelo propuesto. Básicamente, es necesario agregar nuevas restricciones que relacionen el número de camiones y de conductores que pasan por determinados arcos del digrafo.

- *Restricciones de sincronización:*

$$\sum_{v \in V} X_{ve} \leq \sum_{c \in C} Y_{ce} \qquad \forall e \in E^{\text{VIAJ}} \cup E^{\text{R}} \cup E^{\text{E}}$$
$$2 \sum_{v \in V} X_{ve} \geq \sum_{c \in C} Y_{ce} \qquad \forall e \in E^{\text{R}} \cup E^{\text{E}}$$

  donde $E^{\text{R}} = \cup_{p \in P} E^{\text{R},p}$ y $E^{\text{E}} = \cup_{p \in P} E^{\text{E},p}$.

  La primera familia de restricciones dice que en todo arco e de viaje, de recolección y de entrega, el número de camiones que recorren e no puede exceder el número de conductores que lo recorren, ya que de lo contrario habría camiones que no tengan conductores.

  La segunda, dice que en todo arco e de recolección y de entrega, el número de conductores que recorren e no puede exceder dos veces el número de camiones que la recorren, ya que de lo contrario habría conductores sin lugar en los camiones.

Observar que las restricciones propuestas no acotan superiormente el número de conductores sobre los arcos de viaje. Como se ha mencionado anteriormente, siempre que este número exceda el doble de la cantidad de camiones que lo recorren, una cantidad de conductores igual a su diferencia requerirán del servicio de taxis. Este punto será manejado por medio de la función objetivo.

6.1.4. Funciones objetivo

En este problema multiobjetivo se busca minimizar la penalidad por los días de demora en las entregas y los costos asociados a la distancia recorrida por los camiones y por los taxis. Para modelar estos objetivos, se incorporan costos a los arcos de los digrafos.

En este sentido, se definen las funciones costoEntrega, costoCamion, y costoTaxi, que determinan el costo de los arcos respecto a cada uno de los objetivos. Formal-

mente,

$$\text{costoEntrega}(e) = \begin{cases} (\text{dia}(\text{ini}(e)) - dia_p^{\text{E}}) * w_p & \text{si } e \in E^{\text{E},p}, \\ 0 & \text{caso contrario,} \end{cases}$$

$$\text{costoCamion}(e) = \begin{cases} \text{costoViajeCamion}(\text{ori}(e), \text{des}(e)) & \text{si } e \in E^{\text{VIAJ}}, \\ 0 & \text{caso contrario,} \end{cases}$$

$$\text{costoTaxi}(e) = \begin{cases} \text{costoViajeTaxi}(\text{ori}(e), \text{des}(e)) & \text{si } e \in E^{\text{VIAJ}}, \\ 0 & \text{caso contrario.} \end{cases}$$

El costo de entrega de un arco de entrega e asociado a un pedido p es igual a su penalidad w_p multiplicada por el número de días transcurridos desde que la ventana de tiempo de entrega de p abre por primera vez y el instante de tiempo en el que inicia e. El costo de camión y de taxi de un arco de viaje e está dado por el costo de un viaje en camión y en taxi, respectivamente, entre las localidades de origen y de destino de e. Para esto, se definen las funciones auxiliares ori y des que toman un arco y devuelven su localidad de origen y de destino, respectivamente. Por ejemplo, dado un arco $e = (n_{l_1,i_1}, n_{l_2,i_2}) \in E$, entonces $\text{ori}(e) = l_1$ y $\text{des}(e) = l_2$.

Luego, es posible modelar a las funciones objetivo como se muestra a continuación.

- *Minimizar penalidad por días de demora en las entregas:*

  Las restricciones de cumplimiento de pedidos garantizan que cada pedido es realizado exactamente una vez por algún camión. Luego, este objetivo se puede modelar sumando el costo de los arcos de entrega visitados por los camiones.

$$\text{mín} \sum_{e \in E^{\text{E}}} \text{costoEntrega}(e) \sum_{v \in V} X_{ve}$$

- *Minimizar costo por distancia recorrida en camión:*

  Este objetivo se puede modelar de manera muy sencilla sumando el costo de

camión de los arcos de viaje visitados por los camiones.

$$\text{mín} \sum_{e \in E^{\text{VIAJ}}} \text{costoCamion}(e) \sum_{v \in V} X_{ve}$$

- *Minimizar costo por distancia recorrida en taxi:*

  El abordaje de este objetivo tiene una dificultad, ya que en los modelos propuestos no hay variables que representen de manera directa el número de conductores que requieren taxis. Sin embargo, es posible calcularlo a partir de las demás variables.

  Por ejemplo, considerar un arco de viaje e tal que el número de camiones y de conductores que pasan por e es $m \in \mathbb{N}$ y $n \in \mathbb{N}$, respectivamente. Las restricciones de sincronización garantizan que $m \leq n$. Además, dado que en un camión pueden viajar a lo sumo dos conductores, si $n > 2m$ entonces hay exactamente $n - 2m$ conductores que requieren taxis. De lo contrario, ningún taxi es necesario.

  Luego, para modelar este objetivo se introduce una variable $Z_e \in \mathbb{N}_0$ para cada $e \in E^{\text{VIAJ}}$, tal que, en toda solución óptima, Z_e representa el número de conductores que pasan por e y deben viajar en taxi. Estas variables deben estar sujetas a las siguientes restricciones.

  - *Restricciones de activación de variables de taxi:*

    $$\sum_{c \in C} Y_{ce} - 2 \sum_{v \in V} X_{ve} \leq Z_e \qquad \forall e \in E^{\text{VIAJ}}$$

    Estas restricciones garantizan que el valor de Z_e tiene que ser mayor o igual a la diferencia entre el número de conductores y dos veces el número de camiones que pasan por e.

  Finalmente, este objetivo se puede modelar sumando el costo de taxi de los arcos de viaje visitados por los conductores que requieren taxi:

$$\text{mín} \sum_{e \in E^{\text{VIAJ}}} \text{costoTaxi}(e) Z_e$$

Observar que la minimización de este objetivo evita que las variables asuman innecesariamente valores arbitrariamente grandes.

Cada uno de los elementos necesarios para formular el problema mediante PLE ha sido presentado a lo largo del capítulo. Si bien el lector puede deducir la formulación resultante uniendo cada uno de estos elementos, la misma se presenta de forma detallada a continuación a fines de completitud.

6.1.5. Formulación LT–LT

En esta formulación se utiliza el digrafo G_{LT} para modelar tanto las rutas de camiones como la de conductores.

$$\text{(LT–LT)} \quad \text{mín} \sum_{e \in E^{\text{E}}} \text{costoEntrega}(e) \sum_{v \in V} X_{ve} \qquad (f_{\text{ENTR}})$$

$$\text{mín} \sum_{e \in E^{\text{VIAJ}}} \text{costoCamion}(e) \sum_{v \in V} X_{ve} \qquad (f_{\text{VIAJ}})$$

$$\text{mín} \sum_{e \in E^{\text{VIAJ}}} \text{costoTaxi}(e) Z_e \qquad (f_{\text{TAXI}})$$

$$\text{Sujeto a:} \sum_{e \in \Gamma^-(n) \cap E^c} Y_{ce} = \sum_{e \in \Gamma^+(n) \cap E^c} Y_{ce} \qquad \forall c \in C,\ n \in N \setminus \{n_f, n_s\} \qquad (6.1)$$

$$\sum_{\substack{e \in E^{\text{DESC}}: \\ i \leq \text{ini}(e) < i+I}} Y_{ce} \geq \frac{I}{2} \qquad \forall c \in C,\ i \in \{0, \ldots, I.(H-1)\} \qquad (6.2)$$

$$\sum_{h' \in \{h, \ldots, h+6\}} W_{ch'} \geq 1 \qquad \forall c \in C,\ h \in \{0, \ldots, H-7\} \qquad (6.3)$$

$$\sum_{\substack{e \in E^{\text{DESC}}: \\ I.h \leq \text{ini}(e) < I.h+I}} Y_{ce} \geq I.W_{ch} \qquad \forall c \in C,\ h \in \{0, \ldots, H-1\} \qquad (6.4)$$

$$\sum_{e \in \Gamma^-(n) \cap E^v} X_{ve} = \sum_{e \in \Gamma^+(n) \cap E^v} X_{ve} \qquad \forall v \in V,\ n \in N \setminus \{n_f, n_s\} \qquad (6.5)$$

$$\sum_{e \in E^{\text{R},p}} \sum_{v \in V} X_{ve} = 1 \qquad \forall p \in P \qquad (6.6)$$

$$\sum_{e \in E^{\text{R},p}} X_{ve} = \sum_{e \in E^{\text{E},p}} X_{ve} \qquad \forall v \in V,\ p \in P \qquad (6.7)$$

$$\sum_{\substack{e' \in E^{\text{R},p}: \\ \text{fin}(e') \leq \text{ini}(e)}} \sum_{v \in V} X_{ve'} \geq \sum_{v \in V} X_{ve} \qquad \forall p \in P,\ e \in E^{\text{E},p} \qquad (6.8)$$

$$\sum_{p\in P}\ (\sum_{\substack{e\in E^{\mathrm{R},p}:\\ \mathrm{fin}(e)\le i}} X_{ve} - \sum_{\substack{e\in E^{\mathrm{E},p}:\\ \mathrm{fin}(e)\le i}} X_{ve}) \le 1 \qquad \forall v\in V,\ i\in\mathcal{I}^{\mathrm{R}} \tag{6.9}$$

$$\sum_{v\in V} X_{ve} \le \sum_{c\in C} Y_{ce} \qquad \forall e\in E^{\mathrm{VIAJ}}\cup E^{\mathrm{R}}\cup E^{\mathrm{E}} \tag{6.10}$$

$$2\sum_{v\in V} X_{ve} \ge \sum_{c\in C} Y_{ce} \qquad \forall e\in E^{\mathrm{R}}\cup E^{\mathrm{E}} \tag{6.11}$$

$$\sum_{c\in C} Y_{ce} - 2\sum_{v\in V} X_{ve'} \le Z_e \qquad \forall e\in E^{\mathrm{VIAJ}} \tag{6.12}$$

$$X_{ve}\in\{0,1\} \qquad \forall v\in V,\ e\in E^{v} \tag{6.13}$$

$$Y_{ce}\in\{0,1\} \qquad \forall c\in C,\ e\in E^{c} \tag{6.14}$$

$$W_{ch}\in\{0,1\} \qquad \forall c\in C,\ h\in\{0,\dots,H-1\} \tag{6.15}$$

$$Z_e\in\mathbb{N}_0 \qquad \forall e\in E^{\mathrm{VIAJ}} \tag{6.16}$$

En lo que sigue, se presentarán digrafos con más estructura que permitirán relajar la correspondencia entre caminos dirigidos y rutas, a cambio de un mayor número de nodos y/o arcos. Así, algunas restricciones impuestas previamente en el modelo para evitar caminos dirigidos prohibidos se volverán redundantes. El primero de estos digrafos, codificará en los nodos información respecto a si el camión está transportando una mercancía o si se encuentra sin carga alguna, lo que traerá ciertas ventajas en el modelado de las rutas de camión.

6.2. Digrafo LTC – Localidad-Tiempo-Carga

En este digrafo, los nodos son de la forma $n^q_{l,i}$ para cada localidad l, instante de tiempo i y $q\in\{\circ,\bullet\}$, de forma tal que $q=\circ$ si el camión está descargado y $q=\bullet$ si el camión tiene una mercancía cargada.

Formalmente, se define el multidigrafo $G_{\mathrm{LTC}}=(N,E)$. El conjunto N contiene un nodo $n^q_{l,i}$ para cada $l\in L$, $i\in\mathcal{I}$ y $q\in\{\circ,\bullet\}$, y dos nodos especiales, la fuente n_f y el sumidero n_s. En este caso, el multiconjunto de arcos E queda definido por la unión de los siguientes conjuntos:

- E^{DESC} contiene un arco de la forma $(n^q_{l,i}, n^q_{l,i+1})$ para cada $l\in L$, $i\in\mathcal{I}$ con $i\le I.H-1$ y $q\in\{\circ,\bullet\}$.

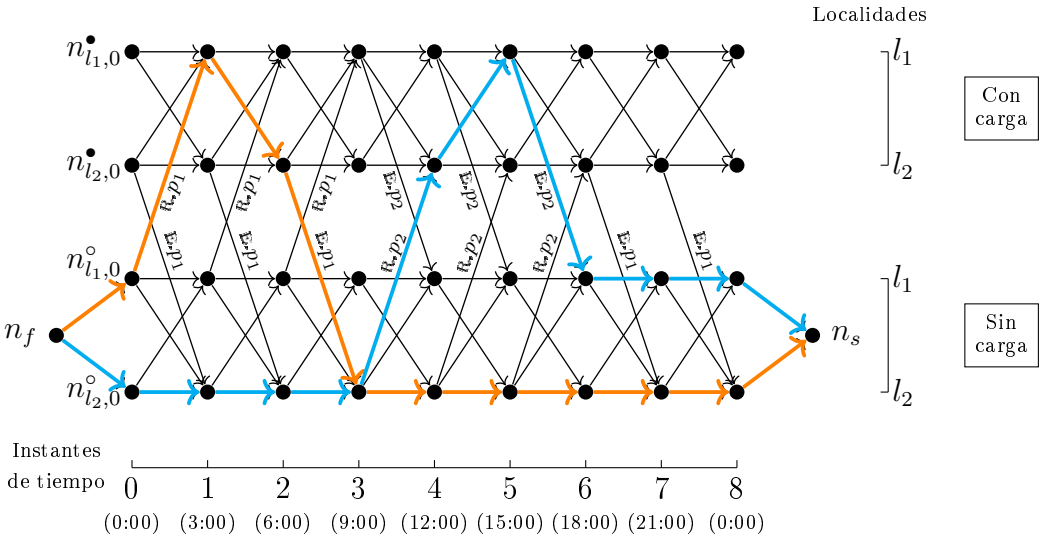


Figura 6.2.1.: Ejemplos para el digrafo LTC – Localidad-Tiempo-Carga.

- E^{VIAJ} contiene un arco de la forma $(n^q_{l_1,i}, n^q_{l_2,i+\Delta})$ para cada par $l_1, l_2 \in L$ de localidades distintas y adyacentes, $i \in \mathcal{I}$ con $i \leq I.H - \Delta$ y $q \in \{\circ, \bullet\}$, donde $\Delta = \text{tiempoViajeCamion}(l_1, l_2)$.
- Para cada $p \in P$, $E^{\text{R},p}$ contiene un arco de la forma $(n^{\circ}_{l^{\text{R}}_p,i}, n^{\bullet}_{l^{\text{R}}_p,i+s^{\text{R}}_p})$ para cada $i \in \mathcal{I}^{\text{R}}_p$.
- Para cada $p \in P$, $E^{\text{E},p}$ contiene un arco de la forma $(n^{\bullet}_{l^{\text{E}}_p,i}, n^{\circ}_{l^{\text{E}}_p,i+s^{\text{E}}_p})$ para cada $i \in \mathcal{I}^{\text{E}}_p$.
- E^{FUEN} contiene un arco de la forma $(n_f, n^{\circ}_{l,0})$ para cada $l \in L$.
- E^{SUMI} contiene un arco de la forma $(n^{\circ}_{l,I.H}, n_s)$ para cada $l \in L$.

A continuación, se provee un ejemplo de esta construcción.

Ejemplo 6.2.1. En la Figura 6.2.1, se muestra el digrafo G_{LTC} que resulta de aplicar esta construcción al Ejemplo 1.2.1 y a las rutas de camiones que propone. En este caso, las dos filas superiores de nodos tienen superíndice $\bullet$ (con carga), mientras que las dos inferiores tienen superíndice $\circ$ (sin carga). Además, los arcos de recolección y de entrega conectan ahora nodos con diferente superíndice. Los (n_f, n_s)-caminos dirigidos se corresponden con las rutas de los camiones v_1 (naranja) y v_2 (cyan).

Para modelar rutas de camiones en G_{LTC} usando PLE, se sigue un procedimiento muy similar al propuesto para G_{LT}. Esto es, se define una variable binaria X_{ve} para todo $v \in V$ y $e \in E^v$ tal que $X_{ve} = 1$ si y solo si el camión v pasa por el arco e. A continuación, se verá que algunas de las restricciones del modelo anterior queden implicadas debido a la mayor estructura que posee este nuevo digrafo.

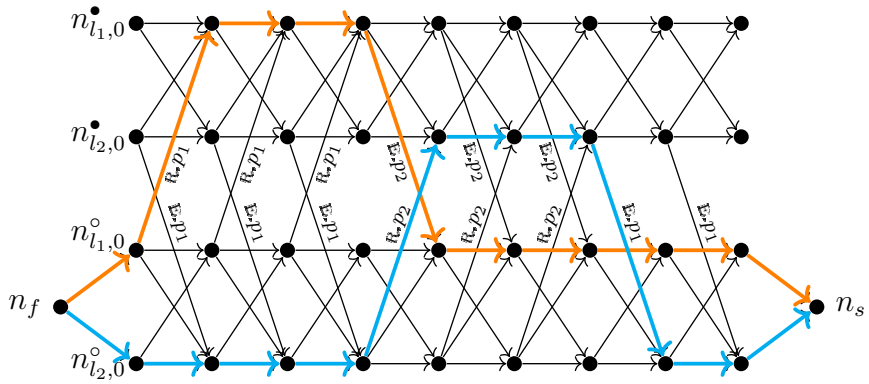
Es fácil ver que la estructura de G_{LTC} garantiza que para todo (n_f, n_s)-camino dirigido, los arcos de recolección y de entrega visitados ocurren en el siguiente orden:

$$\text{R. } p_{i_1},\ \text{E. } p_{i_2},\ \ldots,\ \text{R. } p_{i_{2n-1}},\ \text{E. } p_{i_{2n}}$$

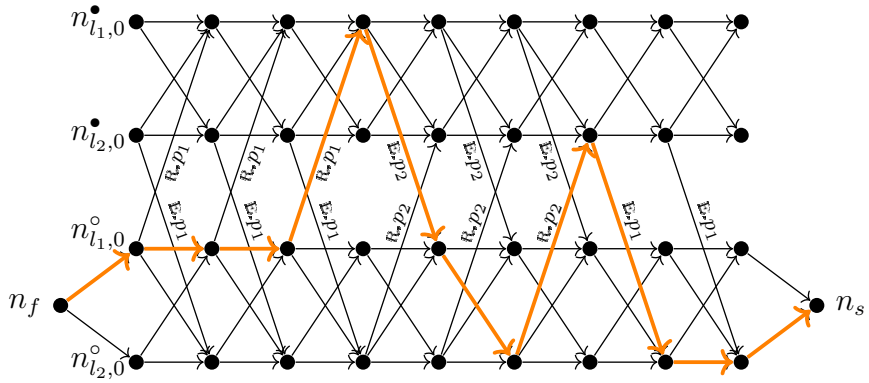
con $p_{i_1}, \ldots, p_{i_{2n}} \in P$ y $n \in \mathbb{N}_0$. Es decir, el camino ya garantiza alternancia entre arcos de recolección y de entrega. Esto se debe básicamente a que la fuente y el sumidero sólo se conectan con nodos con superíndice $\circ$ (sin carga), mientras que los arcos de recolección y de entrega son los únicos que conectan nodos de superíndice $\circ$ y $\bullet$ entre sí (mas específicamente, de $\circ$ a $\bullet$ para los de recolección, y de $\bullet$ a $\circ$ para los de entrega). En consecuencia, las restricciones de capacidad se vuelven redundantes en este modelo. Sin embargo, nada garantiza que los arcos de recolección y de entrega estén emparejados, es decir, que un mismo camión realice la carga y la descarga del pedido, ni que la carga preceda a su descarga; como puede observarse en el siguiente ejemplo.

Ejemplo 6.2.2. Se considera el digrafo G_{LTC} que resulta del Ejemplo 1.2.1 y se sigue la misma codificación de variables que en el Ejemplo 6.1.2. En las Figuras 6.2.2a y 6.2.2b se muestran ejemplos de (n_f, n_s)-caminos dirigidos que no se corresponden con rutas de camión, los mismos son análogos a los de las Figuras 6.1.2a y 6.1.2b pero adaptados a G_{LTC}.

Por lo tanto, las variables de este modelo deben estar sujetas a todas las restricciones del modelo anterior, salvo las de capacidad, con algunas pequeñas modificaciones, las cuales se comentan a continuación. Dado que se usarán digrafos distintos para modelar las rutas de camiones (G_{LTC}) y de conductores (G_{LT}), a partir de este punto y cuando sea necesario evitar ambigüedades, se renombrarán los conjuntos de nodos y de arcos de los mismos de la siguiente forma: $G_{\text{LT}} = (N_{\text{LT}}, E_{\text{LT}})$, $G_{\text{LTC}} = (N_{\text{LTC}}, E_{\text{LTC}})$.



(a) Se incumplen las restricciones de emparejamiento.



(b) Se incumplen las restricciones de precedencia.

Figura 6.2.2.: Ejemplos de (n_f, n_s)-caminos dirigidos prohibidos en G_{LTC}.

- *Restricciones de sincronización:*

  Es evidente que G_{LT} y G_{LTC} tienen conjuntos de nodos y de arcos diferentes. Por ejemplo, dado un pedido p y un arco $(n_{l,i_1}, n_{l,i_2}) \in E^{\text{R},p}$ de G_{LT}, el arco que representa la *misma* transición en G_{LTC} es de la forma $(n^{\circ}_{l,i_1}, n^{\bullet}_{l,i_2})$. Más aún, un arco de G_{LT} puede corresponderse con más de un arco en G_{LTC}. Por ejemplo, dado un arco $(n_{l_1,i_1}, n_{l_2,i_2}) \in E^{\text{VIAJ}}$ de G_{LT}, existen dos arcos que representan la *misma* transición en G_{LTC}: $(n^{q}_{l_1,i_1}, n^{q}_{l_2,i_2})$ con $q \in \{\circ, \bullet\}$. Es decir, un conductor viajando de l_1 a l_2 en el instante i_1 se podría sincronizar con cualquier camión viajando de l_1 a l_2 en el instante i_1 independientemente de su carga. En otras palabras, la carga del camión es irrelevante para la sincronización con los conductores.

  Para expresar las restricciones de sincronización cuando las rutas de conductores y de camiones se modelan con digrafos distintos, es necesario formalizar la correspondencia entre sus arcos. A continuación se define la función, $\text{arco}_{\text{LT}\to\text{LTC}}$ que toma un arco de G_{LT} y devuelve el conjunto de arcos de G_{LTC} que representan la *misma* transición:

$$\text{arco}_{\text{LT}\to\text{LTC}}(e) =$$

$$\begin{cases} \{e' \in E^{\text{DESC}}_{\text{LTC}} : e' = (n^{q}_{l,i_1}, n^{q}_{l,i_2}) \wedge q \in \{\circ, \bullet\}\} & \text{si } e = (n_{l,i_1}, n_{l,i_2}) \in E^{\text{DESC}}_{\text{LT}} \\ \{e' \in E^{\text{VIAJ}}_{\text{LTC}} : e' = (n^{q}_{l_1,i_1}, n^{q}_{l_2,i_2}) \wedge q \in \{\circ, \bullet\}\} & \text{si } e = (n_{l_1,i_1}, n_{l_2,i_2}) \in E^{\text{VIAJ}}_{\text{LT}} \\ \{e' \in E^{\text{R},p}_{\text{LTC}} : e' = (n^{\circ}_{l,i_1}, n^{\bullet}_{l,i_2})\} & \text{si } e = (n_{l,i_1}, n_{l,i_2}) \in E^{\text{R},p}_{\text{LT}} \\ \{e' \in E^{\text{E},p}_{\text{LTC}} : e' = (n^{\bullet}_{l,i_1}, n^{\circ}_{l,i_2})\} & \text{si } e = (n_{l,i_1}, n_{l,i_2}) \in E^{\text{E},p}_{\text{LT}} \\ \{e' \in E^{\text{FUEN}}_{\text{LTC}} : e' = (n_f, n^{\circ}_{l,0})\} & \text{si } e = (n_f, n_{l,0}) \in E^{\text{FUEN}}_{\text{LT}} \\ \{e' \in E^{\text{SUMI}}_{\text{LTC}} : e' = (n^{\circ}_{l,I.H}, n_s)\} & \text{si } e = (n_{l,I.H}, n_s) \in E^{\text{SUMI}}_{\text{LT}} \end{cases}$$

  Así, estas restricciones se pueden expresar como:

$$\sum_{\substack{e' \in \\ \text{arco}_{\text{LT}\to\text{LTC}}(e)}} \sum_{v \in V} X_{ve'} \leq \sum_{c \in C} Y_{ce} \qquad \forall e \in E^{\text{VIAJ}}_{\text{LT}} \cup E^{\text{R}}_{\text{LT}} \cup E^{\text{E}}_{\text{LT}}$$

$$2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LT}\to\text{LTC}}(e)}} \sum_{v \in V} X_{ve'} \geq \sum_{c \in C} Y_{ce} \qquad \forall e \in E^{\text{R}}_{\text{LT}} \cup E^{\text{E}}_{\text{LT}}$$

- *Restricciones de activación de variables de taxi:*

  Estas restricciones también deben adaptarse en la misma línea que las anteriores, esto es:

$$\sum_{c\in C} Y_{ce} - 2\sum_{\substack{e'\in \\ \text{arco}_{\text{LT}\to\text{LTC}}(e)}} \sum_{v\in V} X_{ve'} \leq Z_e \qquad \forall e \in E_{\text{LT}}^{\text{VIAJ}}$$

Para finalizar con la descripción de G_{LTC}, se presentará la formulación de PLE resultante de modelar las rutas de camiones con este digrafo.

6.2.1. Formulación LTC–LT

En esta formulación se utiliza el digrafo G_{LTC} para modelar las rutas de camiones y G_{LT} para las de conductores.

$$\begin{aligned}
\text{(LTC–LT)} \quad \text{mín} \sum_{e\in E_{\text{LTC}}^{\text{E}}} \text{costoEntrega}(e) \sum_{v\in V} X_{ve} \qquad & (f_{\text{ENTR}}) \\
\text{mín} \sum_{e\in E_{\text{LTC}}^{\text{VIAJ}}} \text{costoCamion}(e) \sum_{v\in V} X_{ve} \qquad & (f_{\text{VIAJ}}) \\
\text{mín} \sum_{e\in E_{\text{LT}}^{\text{VIAJ}}} \text{costoTaxi}(e) Z_e \qquad & (f_{\text{TAXI}})
\end{aligned}$$

$$\text{Sujeto a:} \sum_{e\in\Gamma^-(n)\cap E_{\text{LT}}^{c}} Y_{ce} = \sum_{e\in\Gamma^+(n)\cap E_{\text{LT}}^{c}} Y_{ce} \qquad \forall c \in C,\ n \in N_{\text{LT}} \setminus \{n_f, n_s\} \qquad (6.17)$$

$$\sum_{\substack{e\in E_{\text{LT}}^{\text{DESC}}: \\ i\leq \text{ini}(e)<i+I}} Y_{ce} \geq \frac{I}{2} \qquad \forall c \in C,\ i \in \{0,\ldots,I.(H-1)\} \qquad (6.18)$$

$$\sum_{h'\in\{h,\ldots,h+6\}} W_{ch'} \geq 1 \qquad \forall c \in C,\ h \in \{0,\ldots,H-7\} \qquad (6.19)$$

$$\sum_{\substack{e\in E_{\text{LT}}^{\text{DESC}}: \\ I.h\leq \text{ini}(e)<I.h+I}} Y_{ce} \geq I.W_{ch} \qquad \forall c \in C,\ h \in \{0,\ldots,H-1\} \qquad (6.20)$$

$$\sum_{e\in\Gamma^-(n)\cap E_{\text{LTC}}^{v}} X_{ve} = \sum_{e\in\Gamma^+(n)\cap E_{\text{LTC}}^{v}} X_{ve} \qquad \forall v \in V,\ n \in N_{\text{LTC}} \setminus \{n_f, n_s\} \qquad (6.21)$$

$$\sum_{e\in E_{\text{LTC}}^{\text{R},p}} \sum_{v\in V} X_{ve} = 1 \qquad \forall p \in P \qquad (6.22)$$

$$\sum_{e \in E_{\text{LTC}}^{\text{R},p}} X_{ve} = \sum_{e \in E_{\text{LTC}}^{\text{E},p}} X_{ve} \qquad \forall v \in V,\ p \in P \tag{6.23}$$

$$\sum_{\substack{e' \in E_{\text{LTC}}^{\text{R},p}: \\ \text{fin}(e') \leq \text{ini}(e)}} \sum_{v \in V} X_{ve'} \geq \sum_{v \in V} X_{ve} \qquad \forall p \in P,\ e \in E_{\text{LTC}}^{\text{E},p} \tag{6.24}$$

$$\sum_{\substack{e' \in \\ \text{arco}_{\text{LT} \to \text{LTC}}(e)}} \sum_{v \in V} X_{ve'} \leq \sum_{c \in C} Y_{ce} \qquad \forall e \in E_{\text{LT}}^{\text{VIAJ}} \cup E_{\text{LT}}^{\text{R}} \cup E_{\text{LT}}^{\text{E}} \tag{6.25}$$

$$2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LT} \to \text{LTC}}(e)}} \sum_{v \in V} X_{ve'} \geq \sum_{c \in C} Y_{ce} \qquad \forall e \in E_{\text{LT}}^{\text{R}} \cup E_{\text{LT}}^{\text{E}} \tag{6.26}$$

$$\sum_{c \in C} Y_{ce} - 2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LT} \to \text{LTC}}(e)}} \sum_{v \in V} X_{ve'} \leq Z_e \qquad \forall e \in E_{\text{LT}}^{\text{VIAJ}} \tag{6.27}$$

$$X_{ve} \in \{0,1\} \qquad \forall v \in V,\ e \in E_{\text{LTC}}^{v} \tag{6.28}$$

$$Y_{ce} \in \{0,1\} \qquad \forall c \in C,\ e \in E_{\text{LT}}^{c} \tag{6.29}$$

$$W_{ch} \in \{0,1\} \qquad \forall c \in C,\ h \in \{0, \ldots, H-1\} \tag{6.30}$$

$$Z_e \in \mathbb{N}_0 \qquad \forall e \in E_{\text{LT}} \tag{6.31}$$

En el próximo digrafo, los nodos codifican aún más información, pues no sólo indican si el camión está cargado, sino que también permitirán identificar el pedido que está siendo transportado. Nuevamente, esta mayor estructura brindará beneficios a lo hora de modelar rutas de camiones.

6.3. Digrafo LTP – Localidad-Tiempo-Pedido

En este digrafo, para cada localidad l e instante de tiempo i, los nodos son de la forma $n_{l,i}^{\circ}$ si el camión está descargado, o bien de la forma $n_{l,i}^{p}$ si está cargado con el pedido p. Formalmente, se define el digrafo $G_{\text{LTP}} = (N, E)$, donde el conjunto N contiene un nodo $n_{l,i}^{p}$ para cada $l \in L$, $i \in \mathcal{I}$ y $p \in P \cup \{\circ\}$, y dos nodos especiales, la fuente n_f y el sumidero n_s. El conjunto de arcos E queda definido por la unión de los siguientes conjuntos:

- E^{DESC} contiene un arco de la forma $(n_{l,i}^{p}, n_{l,i+1}^{p})$ para cada $l \in L$, $i \in \mathcal{I}$ con $i \leq I.H - 1$ y $p \in P \cup \{\circ\}$.
- E^{VIAJ} contiene un arco de la forma $(n_{l_1,i}^{p}, n_{l_2,i+\Delta}^{p})$ para cada par $l_1, l_2 \in L$ de localidades adyacentes y distintas, $i \in \mathcal{I}$ con $i \leq I.H - \Delta$ y $p \in P \cup \{\circ\}$,

donde $\Delta = \text{tiempoViajeCamion}(l_1, l_2)$.

- Para cada $p \in P$, $E^{\text{R},p}$ contiene un arco de la forma $(n^{\circ}_{l^{\text{R}}_p,i}, n^{p}_{l^{\text{R}}_p,i+s^{\text{R}}_p})$ para cada $i \in \mathcal{I}^{\text{R}}_p$.
- Para cada $p \in P$ $E^{\text{E},p}$, contiene un arco de la forma $(n^{p}_{l^{\text{E}}_p,i}, n^{\circ}_{l^{\text{E}}_p,i+s^{\text{E}}_p})$ para cada $i \in \mathcal{I}^{\text{E}}_p$.
- E^{FUEN} contiene un arco de la forma $(n_f, n^{\circ}_{l,0})$ para cada $l \in L$.
- E^{SUMI} contiene un arco de la forma $(n^{\circ}_{l,I.H}, n_s)$ para cada $l \in L$.

A continuación, se ilustra con un ejemplo esta construcción.

Ejemplo 6.3.1. En la Figura 6.3.1, se muestra el digrafo G_{LTP} que resulta de aplicar esta construcción al Ejemplo 1.2.1 y a las rutas de camiones que propone. En este caso, las dos filas superiores de nodos tienen superíndice p_1 (con carga p_1), las dos filas inferiores tienen superíndice p_2 (con carga p_2) y las dos filas centrales tienen superíndice $\circ$ (sin carga). Además, los arcos de recolección y de entrega asociados a un pedido p conectan ahora nodos de superíndice p con nodos de superíndice $\circ$. Los (n_f, n_s)-caminos dirigidos se corresponden con las rutas de los camiones v_1 (naranja) y v_2 (cyan).

Es fácil ver que la gran estructura que posee G_{LTP} garantiza que para todo (n_f, n_s)-camino dirigido, los arcos de recolección y de entrega visitados ocurren en el siguiente orden:

$$\text{R. } p_{i_1},\ \text{E. } p_{i_1},\ \ldots,\ \text{R. } p_{i_n},\ \text{E. } p_{i_n}$$

con $p_{i_1}, \ldots, p_{i_n} \in P$ y $n \in \mathbb{N}_0$. Es decir, el camino ya garantiza alternancia entre arcos de recolección y de entrega asociados a un mismo pedido. Esto se debe básicamente a que la fuente y el sumidero sólo se conectan con nodos con superíndice $\circ$ (sin carga), que los arcos de recolección asociados a un pedido p son los únicos que conectan nodos de superíndice $\circ$ con nodos de superíndice p, y lo opuesto ocurre con los arcos de entrega de p, y que además un nodo con superíndice $p \in P$ se conecta únicamente con nodos con superíndice $\circ$ o p, pero nunca con aquellos con superíndice en $P \setminus \{p\}$. En consecuencia, las restricciones de emparejamiento, precedencia y capacidad se vuelven redundantes en este modelo. Sin embargo, nada impide que un mismo pedido sea recolectado o entregado más de una vez,

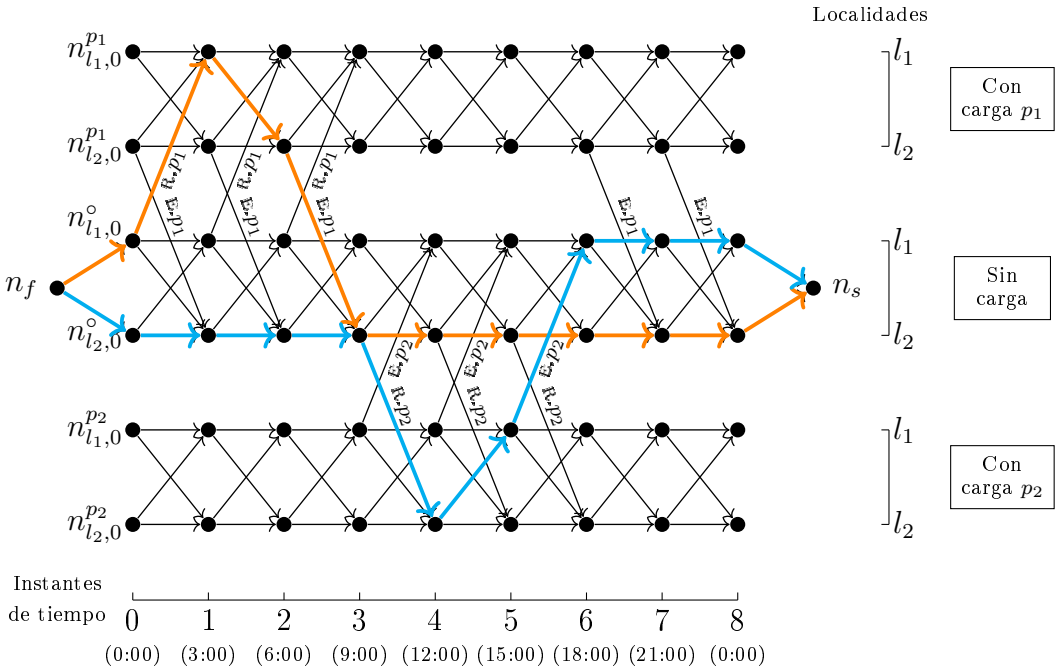


Figura 6.3.1.: Ejemplos para el digrafo LTP – Localidad-Tiempo-Pedido.

aunque esto es fácil de imponer por medio de las restricciones de cumplimiento de pedidos.

Para modelar rutas de camiones en G_{LTP} usando PLE, se podría seguir un procedimiento similar al propuesto para G_{LTC}. Es decir, por medio de una variable X_{ve} para todo $v \in V$ y $e \in E^v$, tal que $X_{ve} = 1$ si y solo si el camión v pasa por el arco e. Sin embargo, este enfoque no aprovecha por completo la mayor estructura que posee este digrafo. Una posible mejora surge de observar que ya no es necesario distinguir a las variables por vehículo, es decir, es suficiente con una variable $X_e \in \mathbb{N}_0$ para todo $e \in E$, tal que el valor que asume representa el número de camiones que pasan por ese arco. El valor que puede asumir X_e está acotado superiormente por el máximo número de camiones que pueden pasar por e. Dicho valor, denotado maxNroCamiones(e), se define como:

$$\text{maxNroCamiones}(e) =$$

$$\begin{cases} |\{v \in V : l_v = l\}| & \text{si } e = (n_f, n^{\circ}_{l,0}) \in E^{\text{FUEN}} \\ |V| & \text{si } e = (n^{\circ}_{l,I.H}, n_s) \in E^{\text{SUMI}} \text{ o } e = (n^{\circ}_{l_1,i_1}, n^{\circ}_{l_2,i_2}) \in E^{\text{DESC}} \cup E^{\text{VIAJ}} \\ 1 & \text{en caso contrario} \end{cases}$$

Es decir, el número de camiones en los arcos que salen de la fuente está acotado superiormente por el número de camiones disponibles en cada localidad inicial. En los arcos que llegan al sumidero y en los que conectan dos nodos sin carga (con superíndice igual a $\circ$), por el número total de camiones. Finalmente, en el resto de los arcos, es decir, los incidentes a un nodo con carga (superíndice distinto de $\circ$), el número de camiones es a lo sumo 1, dado que cada pedido debe ser realizado por exactamente uno de ellos. Por lo tanto, debe valer $X_e \in \{0, \ldots, \text{maxNroCamiones}(e)\}$ para todo $e \in E$.

Estas nuevas variables deben estar sujetas a las siguientes restricciones, varias de las cuales son adaptaciones de las presentadas en los modelos anteriores.

- *Restricciones de conservación de flujo:*

$$\sum_{e \in \Gamma^-(n)} X_e = \sum_{e \in \Gamma^+(n)} X_e \qquad \forall n \in N_{\text{LTP}} \setminus \{n_f, n_s\}.$$

- *Restricciones de cumplimiento de pedidos:*

$$\sum_{e \in E_{\text{LTP}}^{\text{R},p}} X_e = 1 \qquad \forall p \in P.$$

- *Restricciones de sincronización:*

$$\sum_{\substack{e' \in \\ \text{arco}_{\text{LT}\to\text{LTP}}(e)}} X_{e'} \leq \sum_{c \in C} Y_{ce} \qquad \forall e \in E_{\text{LT}}^{\text{VIAJ}} \cup E_{\text{LT}}^{\text{R}} \cup E_{\text{LT}}^{\text{E}}$$

$$2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LT}\to\text{LTP}}(e)}} X_{e'} \geq \sum_{c \in C} Y_{ce} \qquad \forall e \in E_{\text{LT}}^{\text{R}} \cup E_{\text{LT}}^{\text{E}}$$

donde

$$\text{arco}_{\text{LT}\to\text{LTP}}(e) =$$

$$\begin{cases} \{e' \in E_{\text{LTP}}^{\text{DESC}} : e' = (n_{l,i_1}^p, n_{l,i_2}^p) \wedge p \in P \cup \{\circ\}\} & \text{si } e = (n_{l,i_1}, n_{l,i_2}) \in E_{\text{LT}}^{\text{DESC}} \\ \{e' \in E_{\text{LTP}}^{\text{VIAJ}} : e' = (n_{l_1,i_1}^p, n_{l_2,i_2}^p) \wedge p \in P \cup \{\circ\}\} & \text{si } e = (n_{l_1,i_1}, n_{l_2,i_2}) \in E_{\text{LT}}^{\text{VIAJ}} \\ \{e' \in E_{\text{LTP}}^{\text{R},p} : e' = (n_{l,i_1}^\circ, n_{l,i_2}^p)\} & \text{si } e = (n_{l,i_1}, n_{l,i_2}) \in E_{\text{LT}}^{\text{R},p} \\ \{e' \in E_{\text{LTP}}^{\text{E},p} : e' = (n_{l,i_1}^p, n_{l,i_2}^\circ)\} & \text{si } e = (n_{l,i_1}, n_{l,i_2}) \in E_{\text{LT}}^{\text{E},p} \\ \{e' \in E_{\text{LTP}}^{\text{FUEN}} : e' = (n_f, n_{l,0}^\circ)\} & \text{si } e = (n_f, n_{l,0}) \in E_{\text{LT}}^{\text{FUEN}} \\ \{e' \in E_{\text{LTP}}^{\text{SUMI}} : e' = (n_{l,I.H}^\circ, n_s)\} & \text{si } e = (n_{l,I.H}, n_s) \in E_{\text{LT}}^{\text{SUMI}} \end{cases}$$

- *Restricciones de activación de variables de taxi:*

$$\sum_{c \in C} Y_{ce} - 2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LT} \to \text{LTP}}(e)}} X_{e'} \leq Z_e \qquad \forall e \in E_{\text{LT}}^{\text{VIAJ}}$$

- Funciones objetivo:

$$\text{mín} \sum_{e \in E_{\text{LTP}}^{\text{E}}} \text{costoEntrega}(e) X_e$$
$$\text{mín} \sum_{e \in E_{\text{LTP}}^{\text{VIAJ}}} \text{costoCamion}(e) X_e$$

Resta ver cómo se descompone una solución expresada en término de las nuevas variables en (n_f, n_s)-caminos dirigidos. Para ilustrar el procedimiento se recurre al siguiente ejemplo.

Ejemplo 6.3.2. Se considera el digrafo G_{LTP} que resulta del Ejemplo 1.2.1. En la Figura 6.3.2a, se plantea una posible solución para el modelo considerando la siguiente codificación: los arcos e de color magenta tienen su variable asociada $X_e = 1$ y los demás, $X_e = 0$. Existen dos formas de particionar este flujo en (n_f, n_s)-caminos dirigidos, dado que el valor del flujo entrante en el nodo $n_{l_2,3}^\circ$ es igual a 2, y en todos los demás nodos es igual a 1. Una de estas posibilidades se muestra en la ya mencionada Figura 6.3.1. La otra posibilidad se presenta en la Figura 6.3.2b, donde curiosamente el camino dirigido cyan visita únicamente arcos de descanso (salvo el que sale de la fuente y el que llega al sumidero). Es evidente que las dos posibles descomposiciones

generan rutas de camiones distintas, aunque ambas son válidas y tienen el mismo costo, con lo cual a priori es indiferente cuál de ellas elegir.

La conclusión de este ejemplo se generaliza a cualquier instancia, es decir, siempre se puede particionar el flujo de una solución en (n_f, n_s)-caminos dirigidos y cada posible partición se corresponde con un plan de rutas válido para los camiones. En particular, las diferentes alternativas para la partición se originan cuando los caminos dirigidos se cruzan en un mismo nodo, es decir, cuando el flujo entrante/saliente en un nodo tiene un valor mayor a 1. En el modelo, esto puede ocurrir únicamente en nodos con superíndice igual a $\circ$ y allí las rutas se pueden intercambiar libremente porque los camiones están descargados. Por el contrario, cuando los camiones están cargados, las rutas no se deben intercambiar debido a que se violarían las restricciones de emparejamiento. Este último escenario no puede ocurrir en el modelo ya que el flujo entrante/saliente en los nodos con carga siempre es menor o igual a 1.

A continuación, se presenta la formulación de PLE que resulta de aplicar este digrafo para modelar las rutas de camiones.

6.3.1. Formulación LTP–LT

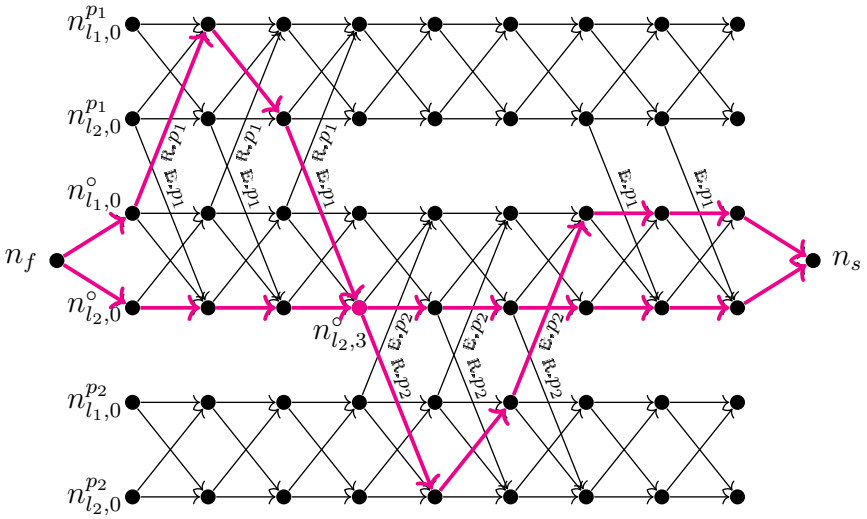
En esta formulación se utiliza el digrafo G_{LTP} para modelar las rutas de camiones y G_{LT} para las de conductores.

$$\text{(LTP–LT)} \quad \text{mín} \sum_{e \in E^{\text{E}}_{\text{LTP}}} \text{costoEntrega}(e) X_e \qquad (f_{\text{ENTR}})$$

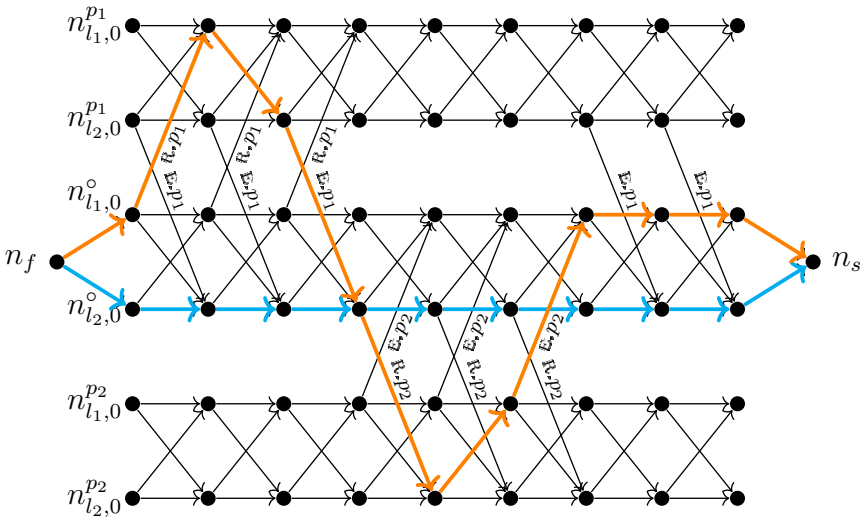
$$\text{mín} \sum_{e \in E^{\text{VIAJ}}_{\text{LTP}}} \text{costoCamion}(e) X_e \qquad (f_{\text{VIAJ}})$$

$$\text{mín} \sum_{e \in E^{\text{VIAJ}}_{\text{LT}}} \text{costoTaxi}(e) Z_e \qquad (f_{\text{TAXI}})$$

$$\text{Sujeto a:} \sum_{e \in \Gamma^-(n) \cap E^c_{\text{LT}}} Y_{ce} = \sum_{e \in \Gamma^+(n) \cap E^c_{\text{LT}}} Y_{ce} \qquad \forall c \in C,\ n \in N_{\text{LT}} \setminus \{n_f, n_s\} \qquad (6.32)$$



(a) Una posible solución.



(b) Dos posibles (n_f, n_s)-caminos dirigidos.

Figura 6.3.2.: Descomposición de soluciones en (n_f, n_s)-caminos dirigidos.

$$\sum_{\substack{e \in E_{\text{LT}}^{\text{DESC}}: \\ i \leq \text{ini}(e) < i+I}} Y_{ce} \geq \frac{I}{2} \qquad \forall c \in C,\ i \in \{0, \ldots, I.(H-1)\} \tag{6.33}$$

$$\sum_{h' \in \{h, \ldots, h+6\}} W_{ch'} \geq 1 \qquad \forall c \in C,\ h \in \{0, \ldots, H-7\} \tag{6.34}$$

$$\sum_{\substack{e \in E_{\text{LT}}^{\text{DESC}}: \\ I.h \leq \text{ini}(e) < I.h+I}} Y_{ce} \geq I.W_{ch} \qquad \forall c \in C,\ h \in \{0, \ldots, H-1\} \tag{6.35}$$

$$\sum_{e \in \Gamma^-(n)} X_e = \sum_{e \in \Gamma^+(n)} X_e \qquad \forall n \in N_{\text{LTP}} \setminus \{n_f, n_s\} \tag{6.36}$$

$$\sum_{e \in E_{\text{LTP}}^{\text{R},p}} X_e = 1 \qquad \forall p \in P \tag{6.37}$$

$$\sum_{\substack{e' \in \\ \text{arco}_{\text{LT} \to \text{LTP}}(e)}} X_{e'} \leq \sum_{c \in C} Y_{ce} \qquad \forall e \in E_{\text{LT}}^{\text{VIAJ}} \cup E_{\text{LT}}^{\text{R}} \cup E_{\text{LT}}^{\text{E}} \tag{6.38}$$

$$2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LT} \to \text{LTP}}(e)}} X_{e'} \geq \sum_{c \in C} Y_{ce} \qquad \forall e \in E_{\text{LT}}^{\text{R}} \cup E_{\text{LT}}^{\text{E}} \tag{6.39}$$

$$\sum_{c \in C} Y_{ce} - 2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LT} \to \text{LTP}}(e)}} X_{e'} \leq Z_e \qquad \forall e \in E_{\text{LT}}^{\text{VIAJ}} \tag{6.40}$$

$$X_e \in \{0, \ldots, \text{maxNroCamiones}(e)\} \qquad \forall e \in E_{\text{LTP}} \tag{6.41}$$

$$Y_{ce} \in \{0, 1\} \qquad \forall c \in C,\ e \in E_{\text{LT}}^{\text{c}} \tag{6.42}$$

$$W_{ch} \in \{0, 1\} \qquad \forall c \in C,\ h \in \{0, \ldots, H-1\} \tag{6.43}$$

$$Z_e \in \mathbb{N}_0 \qquad \forall e \in E_{\text{LT}} \tag{6.44}$$

El próximo digrafo consiste en una alternativa a G_{LT} para el modelado de las rutas de conductores, que hará posible distinguir los conductores que viajan en camión de los que lo hacen en taxi. Esta mayor estructura permitirá expresar de manera más directa el objetivo de minimizar el costo por la distancia recorrida en taxi.

6.4. Digrafo LTX – Variante con arcos de taxi

El digrafo G_{LT} presenta dos grandes desventajas para representar rutas de conductores. Como se ha mencionado anteriormente, para conocer el número de conductores que requieren taxis, es necesario introducir nuevas variables y restricciones al modelo. A su vez, dado que los conductores que se trasladan en camión o en taxi usan los mismos arcos de viaje, se asume que el tiempo de viaje en ambos vehículos es el mismo. Estas cuestiones pueden solucionarse nuevamente agregando más

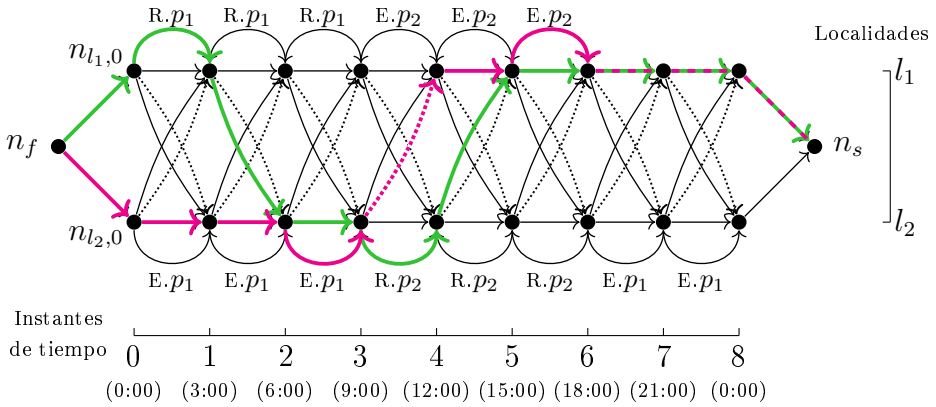


Figura 6.4.1.: Ejemplos para el digrafo LTX – Localidad-Tiempo con arcos de taxi.

estructura al digrafo, a continuación se presenta una variante que adiciona arcos de taxi.

Sea el multidigafo $G_{\text{LTX}} = (N_{\text{LTX}}, E_{\text{LTX}})$ con $N_{\text{LTX}} = N_{\text{LT}}$ y $E_{\text{LTX}} = E_{\text{LT}} \cup E^{\text{TAXI}}$, donde E^{TAXI} es el conjunto de arcos de taxi y modelan que un conductor se desplaza en taxi entre dos localidades. Este conjunto contiene un arco de la forma $(n_{l_1,i}, n_{l_2,i+\Delta})$ para cada par $l_1, l_2 \in L$ de localidades adyacentes y distintas e $i \in \mathcal{I}$ con $i \leq I.H - \Delta$, donde $\Delta = \text{tiempoViajeTaxi}(l_1, l_2)$.

A continuación, se presenta un ejemplo de aplicación de esta construcción.

Ejemplo 6.4.1. En la Figura 6.4.1, se muestra el digrafo G_{LTX} que resulta de aplicar esta construcción al Ejemplo 1.2.1 y a las rutas de conductores que propone. En este caso, los arcos de viaje se dibujan en línea curva continua y los arcos de taxi, en línea curva de puntos. Los (n_f, n_s)-caminos dirigidos coloreados se corresponden con las rutas de los conductores c_1 (verde) y c_2 (magenta).

Para modelar rutas de conductores mediante PLE en G_{LTX}, basta con considerar una variable binaria Y_{ce} para cada $c \in C$ y $e \in E^c_{\text{LTX}}$, tal que $Y_{ce} = 1$ si y sólo si el conductor c pasa por el arco e. Estas variables tienen que estar sujetas a todas las restricciones presentadas para las rutas de conductores en G_{LT}, es decir, las de conservación de flujo y las de descanso de 12 y 24 horas. Respecto a las restricciones de sincronización y a las funciones objetivo, es necesario realizar algunas pequeñas adaptaciones, mencionadas a continuación.

- *Restricciones de sincronización:*

  Dado que en este digrafo los arcos distinguen cuando un conductor se traslada en camión (arcos de viaje) y cuando en taxi (arcos de taxi), es necesario restringir el número de conductores que pasan por un arco de viaje, considerando que no puede superar el doble del número de camiones disponibles para esa transición.

  Así, las restricciones de sincronización se deben adaptar de la siguiente forma:

$$\sum_{v\in V} X_{ve} \leq \sum_{c\in C} Y_{ce} \leq 2\sum_{v\in V} X_{ve} \qquad \forall e \in E^{\text{VIAJ}}_{\text{LTX}} \cup E^{\text{R}}_{\text{LTX}} \cup E^{\text{E}}_{\text{LTX}}$$

$$\sum_{\substack{e'\in \\ \text{arco}_{\text{LTX}\to\text{LTC}}(e)}} \sum_{v\in V} X_{ve'} \leq \sum_{c\in C} Y_{ce} \leq 2\sum_{\substack{e'\in \\ \text{arco}_{\text{LTX}\to\text{LTC}}(e)}} \sum_{v\in V} X_{ve'} \qquad \forall e \in E^{\text{VIAJ}}_{\text{LTX}} \cup E^{\text{R}}_{\text{LTX}} \cup E^{\text{E}}_{\text{LTX}}$$

$$\sum_{\substack{e'\in \\ \text{arco}_{\text{LTX}\to\text{LTP}}(e)}} X_{e'} \leq \sum_{c\in C} Y_{ce} \leq 2\sum_{\substack{e'\in \\ \text{arco}_{\text{LTX}\to\text{LTP}}(e)}} X_{e'} \qquad \forall e \in E^{\text{VIAJ}}_{\text{LTX}} \cup E^{\text{R}}_{\text{LTX}} \cup E^{\text{E}}_{\text{LTX}}$$

  donde la primera familia de restricciones corresponde al caso en que se usa G_{LT} para modelar las rutas de camiones, la segunda a G_{LTC} y la tercera a G_{LTP}. Además, las funciones $\text{arco}_{\text{LTX}\to\text{LTC}}$ y $\text{arco}_{\text{LTX}\to\text{LTP}}$ se definen como:

$$\text{arco}_{\text{LTX}\to\text{LTC}}(e) = \begin{cases} \text{arco}_{\text{LT}\to\text{LTC}}(e) & \text{si } e \notin E^{\text{TAXI}}_{\text{LTX}} \\ \emptyset & \text{caso contrario} \end{cases}$$

$$\text{arco}_{\text{LTX}\to\text{LTP}}(e) = \begin{cases} \text{arco}_{\text{LT}\to\text{LTP}}(e) & \text{si } e \notin E^{\text{TAXI}}_{\text{LTX}} \\ \emptyset & \text{caso contrario} \end{cases}$$

- *Funciones objetivo:*

  Para modelar el objetivo que minimiza el costo asociado a la distancia recorrida en taxi, ya no es necesario la introducción de nuevas variables y restricciones. Este objetivo se puede expresar sencillamente como la suma del costo de taxi de los arcos de taxi visitados por los conductores, es decir,

$$\text{mín} \sum_{e\in E^{\text{TAXI}}_{\text{LTX}}} \text{costoTaxi}(e) \sum_{c\in C} Y_{ce}$$

Para finalizar el capítulo, se detallan las formulaciones de PLE que combinan al nuevo digrafo G_{LTX} para el modelado de las rutas de conductores con los demás digrafos para el modelado de las rutas de camiones.

6.4.1. Formulación LT–LTX

En esta formulación se utiliza el digrafo G_{LT} para modelar las rutas de camiones y G_{LTX} para las de conductores.

$$\text{(LT–LTX)} \quad \text{mín} \sum_{e \in E_{\text{LT}}^{\text{E}}} \text{costoEntrega}(e) \sum_{v \in V} X_{ve} \tag{f_{ENTR}}$$

$$\text{mín} \sum_{e \in E_{\text{LT}}^{\text{VIAJ}}} \text{costoCamion}(e) \sum_{v \in V} X_{ve} \tag{f_{VIAJ}}$$

$$\text{mín} \sum_{e \in E_{\text{LTX}}^{\text{TAXI}}} \text{costoTaxi}(e) \sum_{c \in C} Y_{ce} \tag{f_{TAXI}}$$

$$\text{Sujeto a:} \sum_{e \in \Gamma^-(n) \cap E_{\text{LTX}}^{\text{C}}} Y_{ce} = \sum_{e \in \Gamma^+(n) \cap E_{\text{LTX}}^{\text{C}}} Y_{ce} \qquad \forall c \in C,\ n \in N_{\text{LTX}} \setminus \{n_f, n_s\} \tag{6.45}$$

$$\sum_{\substack{e \in E_{\text{LTX}}^{\text{DESC}}: \\ i \le \text{ini}(e) < i+I}} Y_{ce} \ge \frac{I}{2} \qquad \forall c \in C,\ i \in \{0, \dots, I.(H-1)\} \tag{6.46}$$

$$\sum_{h' \in \{h, \dots, h+6\}} W_{ch'} \ge 1 \qquad \forall c \in C,\ h \in \{0, \dots, H-7\} \tag{6.47}$$

$$\sum_{\substack{e \in E_{\text{LTX}}^{\text{DESC}}: \\ I.h \le \text{ini}(e) < I.h+I}} Y_{ce} \ge I.W_{ch} \qquad \forall c \in C,\ h \in \{0, \dots, H-1\} \tag{6.48}$$

$$\sum_{e \in \Gamma^-(n) \cap E_{\text{LT}}^{v}} X_{ve} = \sum_{e \in \Gamma^+(n) \cap E_{\text{LT}}^{v}} X_{ve} \qquad \forall v \in V,\ n \in N_{\text{LT}} \setminus \{n_f, n_s\} \tag{6.49}$$

$$\sum_{e \in E_{\text{LT}}^{\text{R},p}} \sum_{v \in V} X_{ve} = 1 \qquad \forall p \in P \tag{6.50}$$

$$\sum_{e \in E_{\text{LT}}^{\text{R},p}} X_{ve} = \sum_{e \in E_{\text{LT}}^{\text{E},p}} X_{ve} \qquad \forall v \in V,\ p \in P \tag{6.51}$$

$$\sum_{\substack{e' \in E_{\text{LT}}^{\text{R},p}: \\ \text{fin}(e') \le \text{ini}(e)}} \sum_{v \in V} X_{ve'} \ge \sum_{v \in V} X_{ve} \qquad \forall p \in P,\ e \in E_{\text{LT}}^{\text{E},p} \tag{6.52}$$

$$\sum_{p \in P} \Big(\sum_{\substack{e \in E_{\text{LT}}^{\text{R},p}: \\ \text{fin}(e) \le i}} X_{ve} - \sum_{\substack{e \in E_{\text{LT}}^{\text{E},p}: \\ \text{fin}(e) \le i}} X_{ve} \Big) \le 1 \qquad \forall v \in V,\ i \in \mathcal{I}^{\text{R}} \tag{6.53}$$

$$\sum_{v \in V} X_{ve} \le \sum_{c \in C} Y_{ce} \le 2 \sum_{v \in V} X_{ve} \qquad \forall e \in E_{\text{LTX}}^{\text{VIAJ}} \cup E_{\text{LTX}}^{\text{R}} \cup E_{\text{LTX}}^{\text{E}} \tag{6.54}$$

$$X_{ve} \in \{0,1\} \qquad \forall v \in V,\ e \in E^{v}_{\text{LT}} \tag{6.55}$$

$$Y_{ce} \in \{0,1\} \qquad \forall c \in C,\ e \in E^{c}_{\text{LTX}} \tag{6.56}$$

$$W_{ch} \in \{0,1\} \qquad \forall c \in C,\ h \in \{0,\dots,H-1\} \tag{6.57}$$

6.4.2. Formulación LTC–LTX

En esta formulación se utiliza el digrafo G_{LTC} para modelar las rutas de camiones y G_{LTX} para las de conductores.

$$\text{(LTC–LTX)} \quad \text{mín} \sum_{e \in E^{\text{E}}_{\text{LTC}}} \text{costoEntrega}(e) \sum_{v \in V} X_{ve} \tag{f_{ENTR}}$$

$$\text{mín} \sum_{e \in E^{\text{VIAJ}}_{\text{LTC}}} \text{costoCamion}(e) \sum_{v \in V} X_{ve} \tag{f_{VIAJ}}$$

$$\text{mín} \sum_{e \in E^{\text{TAXI}}_{\text{LTX}}} \text{costoTaxi}(e) \sum_{c \in C} Y_{ce} \tag{f_{TAXI}}$$

Sujeto a:

$$\sum_{e \in \Gamma^{-}(n) \cap E^{c}_{\text{LTX}}} Y_{ce} = \sum_{e \in \Gamma^{+}(n) \cap E^{c}_{\text{LTX}}} Y_{ce} \qquad \forall c \in C,\ n \in N_{\text{LTX}} \setminus \{n_f, n_s\} \tag{6.58}$$

$$\sum_{\substack{e \in E^{\text{DESC}}_{\text{LTX}}: \\ i \le \text{ini}(e) < i+I}} Y_{ce} \ge \frac{I}{2} \qquad \forall c \in C,\ i \in \{0,\dots,I.(H-1)\} \tag{6.59}$$

$$\sum_{h' \in \{h,\dots,h+6\}} W_{ch'} \ge 1 \qquad \forall c \in C,\ h \in \{0,\dots,H-7\} \tag{6.60}$$

$$\sum_{\substack{e \in E^{\text{DESC}}_{\text{LTX}}: \\ I.h \le \text{ini}(e) < I.h+I}} Y_{ce} \ge I.W_{ch} \qquad \forall c \in C,\ h \in \{0,\dots,H-1\} \tag{6.61}$$

$$\sum_{e \in \Gamma^{-}(n) \cap E^{v}_{\text{LTC}}} X_{ve} = \sum_{e \in \Gamma^{+}(n) \cap E^{v}_{\text{LTC}}} X_{ve} \qquad \forall v \in V,\ n \in N_{\text{LTC}} \setminus \{n_f, n_s\} \tag{6.62}$$

$$\sum_{e \in E^{\text{R},p}_{\text{LTC}}} \sum_{v \in V} X_{ve} = 1 \qquad \forall p \in P \tag{6.63}$$

$$\sum_{e \in E^{\text{R},p}_{\text{LTC}}} X_{ve} = \sum_{e \in E^{\text{E},p}_{\text{LTC}}} X_{ve} \qquad \forall v \in V,\ p \in P \tag{6.64}$$

$$\sum_{\substack{e' \in E^{\text{R},p}_{\text{LTC}}: \\ \text{fin}(e') \le \text{ini}(e)}} \sum_{v \in V} X_{ve'} \ge \sum_{v \in V} X_{ve} \qquad \forall p \in P,\ e \in E^{\text{E},p}_{\text{LTC}} \tag{6.65}$$

$$\sum_{\substack{e' \in \\ \text{arco}_{\text{LTX} \to \text{LTC}}(e)}} \sum_{v \in V} X_{ve'} \le \sum_{c \in C} Y_{ce} \le 2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LTX} \to \text{LTC}}(e)}} \sum_{v \in V} X_{ve'} \qquad \forall e \in E^{\text{VIAJ}}_{\text{LTX}} \cup E^{\text{R}}_{\text{LTX}} \cup E^{\text{E}}_{\text{LTX}} \tag{6.66}$$

$$X_{ve} \in \{0,1\} \qquad \forall v \in V,\ e \in E^{v}_{\text{LTC}} \tag{6.67}$$

$$Y_{ce} \in \{0,1\} \qquad \forall c \in C,\ e \in E^c_{\text{LTX}} \tag{6.68}$$

$$W_{ch} \in \{0,1\} \qquad \forall c \in C,\ h \in \{0,\dots,H-1\} \tag{6.69}$$

6.4.3. Formulación LTP–LTX

En esta formulación se utiliza el digrafo G_{LTP} para modelar las rutas de camiones y G_{LTX} para las de conductores.

$$\text{(LTP–LT)} \quad \text{mín} \sum_{e \in E^{\text{E}}_{\text{LTP}}} \text{costoEntrega}(e) X_e \tag{f_{ENTR}}$$

$$\text{mín} \sum_{e \in E^{\text{VIAJ}}_{\text{LTP}}} \text{costoCamion}(e) X_e \tag{f_{VIAJ}}$$

$$\text{mín} \sum_{e \in E^{\text{TAXI}}_{\text{LTX}}} \text{costoTaxi}(e) \sum_{c \in C} Y_{ce} \tag{f_{TAXI}}$$

Sujeto a:

$$\sum_{e \in \Gamma^-(n) \cap E^c_{\text{LTX}}} Y_{ce} = \sum_{e \in \Gamma^+(n) \cap E^c_{\text{LTX}}} Y_{ce} \qquad \forall c \in C,\ n \in N_{\text{LTX}} \setminus \{n_f, n_s\} \tag{6.70}$$

$$\sum_{\substack{e \in E^{\text{DESC}}_{\text{LTX}}: \\ i \le \text{ini}(e) < i+I}} Y_{ce} \ge \frac{I}{2} \qquad \forall c \in C,\ i \in \{0,\dots,I.(H-1)\} \tag{6.71}$$

$$\sum_{h' \in \{h,\dots,h+6\}} W_{ch'} \ge 1 \qquad \forall c \in C,\ h \in \{0,\dots,H-7\} \tag{6.72}$$

$$\sum_{\substack{e \in E^{\text{DESC}}_{\text{LTX}}: \\ I.h \le \text{ini}(e) < I.h+I}} Y_{ce} \ge I.W_{ch} \qquad \forall c \in C,\ h \in \{0,\dots,H-1\} \tag{6.73}$$

$$\sum_{e \in \Gamma^-(n)} X_e = \sum_{e \in \Gamma^+(n)} X_e \qquad \forall n \in N_{\text{LTP}} \setminus \{n_f, n_s\} \tag{6.74}$$

$$\sum_{e \in E^{\text{R},p}_{\text{LTP}}} X_e = 1 \qquad \forall p \in P \tag{6.75}$$

$$\sum_{\substack{e' \in \\ \text{arco}_{\text{LTX}\to\text{LTP}}(e)}} X_{e'} \le \sum_{c \in C} Y_{ce} \le 2 \sum_{\substack{e' \in \\ \text{arco}_{\text{LTX}\to\text{LTP}}(e)}} X_{e'} \qquad \forall e \in E^{\text{VIAJ}}_{\text{LTX}} \cup E^{\text{R}}_{\text{LTX}} \cup E^{\text{E}}_{\text{LTX}} \tag{6.76}$$

$$X_e \in \{0,\dots,\text{maxNroCamiones}(e)\} \qquad \forall e \in E_{\text{LTP}} \tag{6.77}$$

$$Y_{ce} \in \{0,1\} \qquad \forall c \in C,\ e \in E^c_{\text{LTX}} \tag{6.78}$$

$$W_{ch} \in \{0,1\} \qquad \forall c \in C,\ h \in \{0,\dots,H-1\} \tag{6.79}$$

6.5. Reducciones de los digrafos

En esta sección, se mencionan reducciones que pueden hacerse sobre los digrafos, específicamente sobre G_{LTC} y G_{LTP}, con el objetivo de reducir el tamaño de los modelos sin perder soluciones factibles.

En virtud de facilitar la comprensión, las definiciones dadas para G_{LTC} y G_{LTP} incluyeron ciertos nodos y arcos para los cuales es fácil ver que no existen (n_f, n_s)-caminos dirigidos que los recorran y por ende pueden ser removidos sin perder soluciones factibles. Por ejemplo, nodos sin arcos de entrada o de salida (salvo fuente y sumidero), arcos de viaje o descanso en la parte con carga del digrafo (superíndice distinto de $\circ$) que sean anteriores al primer arco de recolección o posteriores al último arco de entrega, entre otros.

El procedimiento de reducción es sencillo y consiste en remover todo nodo $n \in N \setminus \{n_f, n_s\}$ tal que $\Gamma^-(n) = \emptyset$ o $\Gamma^+(n) = \emptyset$ junto a todos los arcos incidentes a n, y repetir este procedimiento hasta que no queden nodos que cumplan esta condición; véase el siguiente ejemplo.

Ejemplo 6.5.1. En la Figura 6.5.1, se muestra el resultado de aplicar la reducción sobre el digrafo G_{LTP} para el Ejemplo 1.2.1. Los nodos y arcos en línea discontinua y de color más claro son los eliminados durante el proceso de reducción.

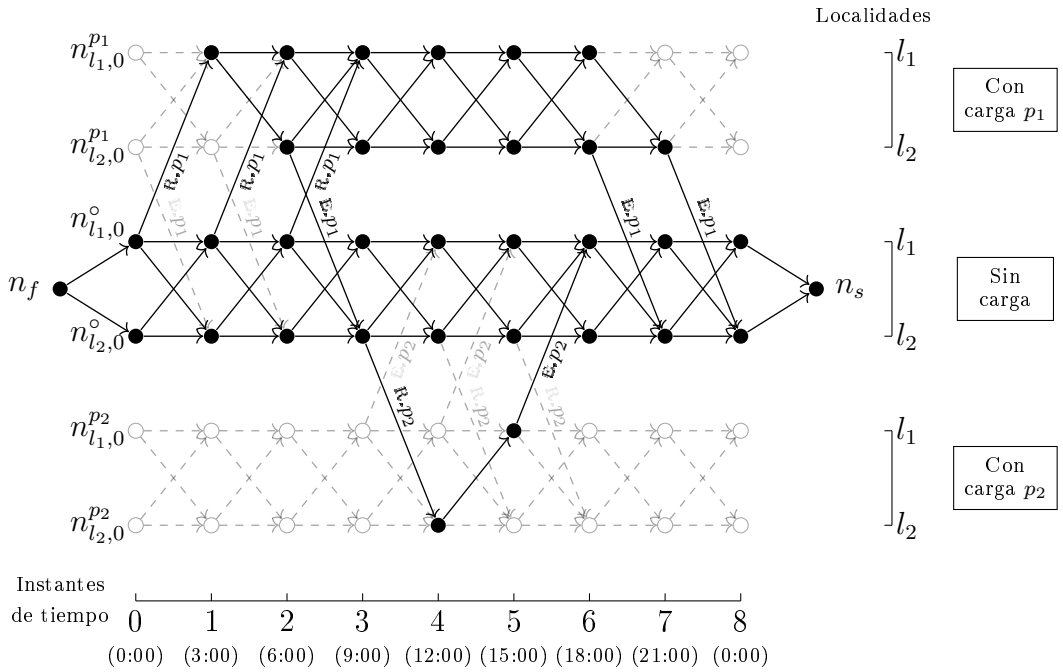


Figura 6.5.1.: Ejemplo de reducción de G_{LTP}.

7. Resolución exacta

Este capítulo se ocupa de proponer formulaciones alternativas para los modelos presentados en el capítulo anterior, caracterizar ciertas estructuras que aparecen recurrentemente en las soluciones fraccionarias de las relajaciones lineales e investigar desigualdades válidas que permitan cortarlas para dar lugar a formulaciones con mayor ajuste. Los resultados presentados serán de utilidad para resolver el problema en estudio de manera más eficiente en el contexto de un algoritmo de optimización exacto.

En general, las restricciones se explicitarán en términos de la primera formulación, i.e. LT-LT, la cual usa el digrafo G_{LT} para modelar tanto las rutas de camiones como de conductores. En la mayoría de los casos, la adaptación a las demás formulaciones es directa y se omite a fin de simplificar la lectura, salvo algunos casos de interés que se encuentran debidamente detallados.

7.1. Formulaciones alternativas

En la primera parte de esta sección, se presentan variantes para las restricciones de precedencia y de descanso semanal. Las mismas, además de ser desigualdades válidas, y por lo tanto candidatas a ser usadas como planos de corte, también tienen la capacidad de reemplazar a las restricciones originales de los modelos, conduciendo a formulaciones más ajustadas del problema en estudio.

Posteriormente, se presentan dos variantes para las restricciones de sincronización. Al reemplazar las restricciones originales por estas variantes propuestas, se modela un problema ligeramente diferente pero *equivalente*, donde las soluciones óptimas del problema alternativo conducen a soluciones óptimas del problema original con mismo valor objetivo.

7.1.1. Restricciones de precedencia

Las restricciones de precedencia fueron descriptas originalmente como:

$$\sum_{\substack{e' \in E^{\mathrm{R},p}: \\ \mathrm{fin}(e') \leq \mathrm{ini}(e)}} \sum_{v \in V} X_{ve'} \geq \sum_{v \in V} X_{ve} \qquad \forall p \in P,\ e \in E^{\mathrm{E},p}, \qquad (\mathrm{PREC}_1)$$

y garantizan que la recolección de un pedido sea anterior a su entrega.

Sean un pedido p y un arco e de entrega de p. Por la estructura del digrafo y las restricciones del modelo, es posible probar que, para toda solución entera, la recolección de p inicia con suficiente anterioridad al inicio de e para que el camión que lo realiza pueda viajar desde la localidad de recolección a la de entrega. Por lo tanto, es posible reforzar estas restricciones limitando los arcos de recolección de p de la siguiente forma:

$$\sum_{\substack{e'' \in E^{\mathrm{R},p}: \\ \mathrm{fin}(e'') \leq \\ \mathrm{ini}(e) - \mathrm{tVC}(l_p^{\mathrm{R}}, l_p^{\mathrm{E}})}} \sum_{v \in V} X_{ve''} \geq \sum_{v \in V} X_{ve} \qquad \forall p \in P,\ e \in E^{\mathrm{E},p}.$$

En estudios preliminares sobre las relajaciones lineales, se observó que los camiones suelen entregar parcialmente los pedidos en múltiples instantes de tiempo, generando que estas desigualdades sean poco ajustadas. Por ejemplo, dado un camión v y dos arcos de entrega e y e' de p, una solución fraccionaria podría asignar $X_{ve} = X_{ve'} = 0,5$. Notando que todo pedido debe ser entregado exactamente una vez en el horizonte de planificación, es posible reforzarlas sumando sobre todos los arcos de entrega de p anteriores a e. Es decir, si $X_{ve} = 1$ en una solución entera, entonces $X_{v'e} = 0$ para todo camión $v' \neq v$ y $X_{v'e'} = 0$ para todo camión v' y arco e' de entrega de p anterior a e. Luego es posible pensar en las siguientes desigualdades:

$$\sum_{\substack{e'' \in E^{\mathrm{R},p}: \\ \mathrm{fin}(e'') \leq \\ \mathrm{ini}(e) - \mathrm{tVC}(l_p^{\mathrm{R}}, l_p^{\mathrm{E}})}} \sum_{v \in V} X_{ve''} \geq \sum_{\substack{e' \in E^{\mathrm{E},p}: \\ \mathtt{fin}(e') \leq \mathtt{fin}(e)}} \sum_{v \in V} X_{ve'} \qquad \forall p \in P,\ e \in E^{\mathrm{E},p}. \qquad (\mathrm{PREC}_2)$$

En particular, si se hace la entrega de p en algún $e' \in E^{\mathrm{E},p}$ anterior a e, entonces la validez de la desigualdad queda implicada por la validez de la restricción de

precedencia asociada a p y e'.

Por lo tanto, reemplazar las restricciones PREC_1 por las PREC_2 define una formulación alternativa para el problema, que además es más ajustada que la original.

7.1.2. Restricciones de descanso

Las restricciones de descanso semanal fueron descriptas como:

$$\sum_{\substack{e \in E^{\text{DESC}}: \\ I.h \leq \text{ini}(e) < I.h+I}} Y_{ce} \geq I.W_{ch} \qquad \forall c \in C,\ h \in \{0, \ldots, H-1\}, \qquad (\text{DESC}_1)$$

y garantizan que un conductor descanse I instantes de tiempo en su día de franco, es decir, que descanse el día entero.

Otra alternativa para estas restricciones consiste en desagregarlas por instante de tiempo, es decir, si un conductor c se encuentra de franco en el día h, entonces se tiene que cumplir que c se encuentra descansando en alguna localidad en todo instante de tiempo i de ese día. Es decir,

$$\sum_{\substack{e \in E^{\text{DESC}}: \\ \text{ini}(e) = i}} Y_{ce} \geq W_{ch} \qquad \begin{array}{r} \forall c \in C,\ h \in \{0, \ldots, H-1\}, \\ i \in \{I.h, \ldots, I.h + I - 1\} \end{array} \qquad (\text{DESC}_2)$$

Nuevamente, reemplazar las restricciones DESC_1 por las DESC_2 define una formulación alternativa para el problema, que además es más ajustada que la original.

7.1.3. Restricciones de sincronización

Las restricciones de sincronización fueron presentadas como:

$$\begin{aligned} \sum_{v \in V} X_{ve} &\leq \sum_{c \in C} Y_{ce} && \forall e \in E^{\text{VIAJ}} \cup E^{\text{R}} \cup E^{\text{E}}, \\ 2 \sum_{v \in V} X_{ve} &\geq \sum_{c \in C} Y_{ce} && \forall e \in E^{\text{R}} \cup E^{\text{E}}. \end{aligned} \qquad (\text{SINC}_1)$$

y garantizan, entre otras cosas, que por los arcos de recolección y entrega pasen al menos uno y a lo sumo dos conductores cuando pasa un camión (más de un camión no puede pasar debido a las restricciones de cumplimiento de pedidos).

De acuerdo a los requerimientos del problema, no es necesario que dos conductores participen simultáneamente en la recolección o entrega de un pedido. Es decir, si en una recolección participan dos conductores, puede pensarse que uno de ellos se queda descansando fuera del camión mientras el otro realiza la carga del pedido. La nueva solución es claramente factible y tiene el mismo costo. Por lo tanto, una alternativa para estas restricciones es:

$$\begin{aligned} \sum_{v \in V} X_{ve} &\leq \sum_{c \in C} Y_{ce} && \forall e \in E^{\text{VIAJ}} \\ \sum_{v \in V} X_{ve} &= \sum_{c \in C} Y_{ce} && \forall e \in E^{\text{R}} \cup E^{\text{E}}. \end{aligned} \tag{SINC$_2$}$$

Por otro lado, en el hipotético caso de que en la recolección de un pedido pudiesen participar más de dos conductores, también es posible pensar que sólo dos de ellos hacen la recolección mientras los demás descansan fuera del camión (y análogamente con las entregas). La nueva solución es claramente factible y tiene el mismo costo. Por lo tanto, otra alternativa es eliminar por completo las restricciones que acotan superiormente al número de conductores que pasan por un arco de recolección o entrega, quedando:

$$\sum_{v \in V} X_{ve} \leq \sum_{c \in C} Y_{ce} \qquad \forall e \in E^{\text{VIAJ}} \cup E^{\text{R}} \cup E^{\text{E}}. \tag{SINC$_3$}$$

Al reemplazar las restricciones SINC$_1$ por las SINC$_2$ o por las SINC$_3$, se obtienen formulaciones para problemas distintos al original. El primero es un problema más restringido, en el cual las tripulaciones que ejecutan las recolecciones y las entregas deben estar constituidas por un único conductor, pero donde una solución óptima es también óptima para el problema original. Mientras que el segundo es un problema más relajado, en el cual las tripulaciones que ejecutan las recolecciones y las entregas pueden estar constituidas por más de dos conductores, pero donde una solución óptima del problema relajado conduce a una solución óptima del problema original con el mismo valor objetivo.

7.2. Desigualdades válidas

En esta sección, se caracterizan ciertas estructuras que aparecen frecuentemente en las soluciones fraccionarias. Además, se muestran diferentes desigualdades válidas que permiten cortarlas, las cuales se pueden clasificar en tres grandes grupos según la estructura que prohíben.

- **Restricciones de viaje recolección-entrega**. Fuerzan a que los camiones pasen por al menos un arco de viaje con origen en la localidad de recolección y por al menos uno con destino a la localidad de entrega por cada pedido que transporten. Dichos arcos pueden ser diferentes si el camión pasa por alguna localidad intermedia al transportar el pedido y, por el contrario, son iguales si lo hace de forma directa.

- **Restricciones de secuenciación de pedidos**. Imponen cotas inferiores en el tiempo requerido para que un conjunto de pedidos pueda ser ejecutado secuencialmente por un mismo camión.

- **Restricciones de viaje en taxi**. En el caso de estudio pueden existir camiones que inicien en una localidad sin conductores. Estas restricciones garantizan que si alguno de dichos camiones es usado para transportar un pedido, entonces es necesario que algún conductor viaje a esa localidad inicial en algún otro camión o en taxi. Particularmente en instancias con un único camión, la única posibilidad es que el conductor viaje en taxi.

A continuación, se estudia cada una de estas familias con mayor profundidad.

7.2.1. Restricciones de viaje recolección-entrega

Existen determinados viajes que los camiones están obligados a realizar en cualquier solución entera. En particular, para todo pedido, el camión que lo realiza debe viajar de la localidad de recolección a la de entrega, aunque no es necesario que lo haga de forma directa. Sin embargo, esta propiedad no siempre es verificada por las soluciones fraccionarias, como lo ilustra el siguiente ejemplo.

Ejemplo 7.2.1. Se considera una instancia del problema que involucra dos localidades l_1 y l_2 y un horizonte de planificación de 1 día discretizado cada 3 horas. El tiempo de viaje entre ambas localidades es de 3 horas (1 instante de tiempo). Se dispone de un único camión v_1 en la localidad inicial l_1 y de suficientes conductores para cumplir con dos pedidos p_1 y p_2, definidos con los siguientes atributos.

Pedido p_1:

$$dia^{\mathrm{R}}_{p_1} = dia^{\mathrm{E}}_{p_1} = 0$$
$$[a^{\mathrm{R}}_{p_1}, b^{\mathrm{R}}_{p_1}] = [1, 4] \text{ (3:00-12:00)}$$
$$[a^{\mathrm{E}}_{p_1}, b^{\mathrm{E}}_{p_1}] = [3, 6] \text{ (9:00-18:00)}$$
$$l^{\mathrm{R}}_{p_1} = l_1$$
$$l^{\mathrm{E}}_{p_1} = l_2$$
$$s^{\mathrm{R}}_{p_1} = s^{\mathrm{E}}_{p_1} = 1 \text{ (3 h)}$$

Pedido p_2:

$$dia^{\mathrm{R}}_{p_2} = dia^{\mathrm{E}}_{p_2} = 0$$
$$[a^{\mathrm{R}}_{p_2}, b^{\mathrm{R}}_{p_2}] = [1, 4] \text{ (3:00-12:00)}$$
$$[a^{\mathrm{E}}_{p_2}, b^{\mathrm{E}}_{p_2}] = [3, 6] \text{ (9:00-18:00)}$$
$$l^{\mathrm{R}}_{p_2} = l_2$$
$$l^{\mathrm{E}}_{p_2} = l_1$$
$$s^{\mathrm{R}}_{p_2} = s^{\mathrm{E}}_{p_2} = 1 \text{ (3 h)}$$

En toda solución entera, v_1 debe pasar por al menos un arco de viaje de l_1 a l_2 (para entregar a p_1) y por al menos uno de l_2 a l_1 (para entregar a p_2). Esto implica que el flujo de v_1 por dichos arcos debe sumar a los sumo 2, quedando la función objetivo f_{VIAJ} acotada inferiormente por:

$$\text{costoViajeCamion}(l_1, l_2) + \text{costoViajeCamion}(l_2, l_1).$$

En la Figura 7.2.1, se muestra una posible solución fraccionaria para la ruta de v_1 en el modelo con el digrafo G_{LT}, donde la variable $X_{v_1 e}$ tiene valor 1 para los arcos e en línea continua naranja, valor 0,5 para los arcos en línea de puntos naranja y valor 0 para los restantes. Puede comprobarse fácilmente que esta solución satisface todas las restricciones de camiones del modelo, pero que v_1 pasa únicamente por un arco de viaje cuyo flujo es de tan solo 0,5. Por lo tanto, su valor objetivo respecto a f_{VIAJ} es:

$$0,5 * \text{costoViajeCamion}(l_1, l_2).$$

La existencia de soluciones fraccionarias como las de este ejemplo hace que el valor óptimo de la relajación lineal sea una cota inferior de muy mala calidad del valor

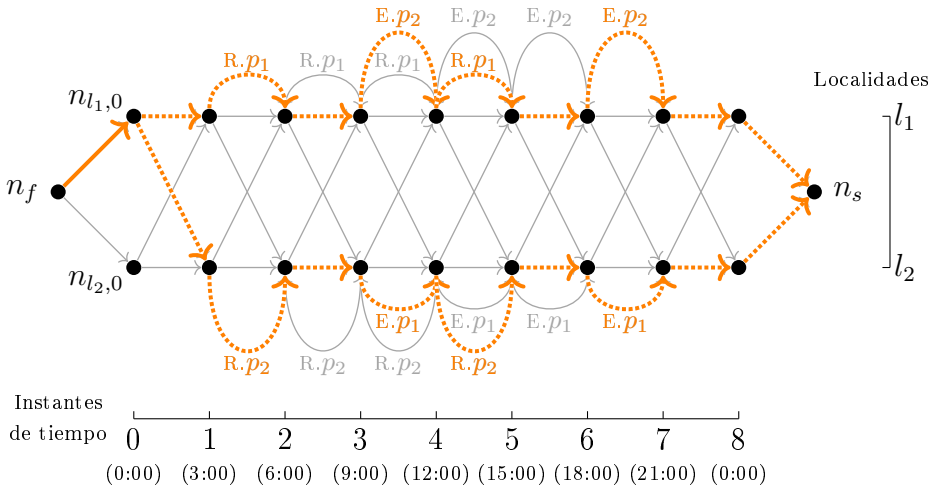


Figura 7.2.1.: Ruta de camión fraccionaria en el digrafo G_{LT}.

óptimo entero. Además, el mismo inconveniente se presenta en el modelo con el digrafo G_{LTC}. Por el contrario, este tipo de soluciones fraccionarias no pueden existir en el modelo con el digrafo G_{LTP}, dado que por su construcción todo camino debe alternar arcos de recolección y entrega emparejados a un mismo pedido. En esta sección, se presentan tres posibles alternativas para cortar soluciones fraccionarias de este tipo.

Sean una solución entera, una localidad l y un camión v. Por la construcción del digrafo y las restricciones del modelo, se puede probar que v pasa por al menos tantos arcos de viaje con origen en l como pedidos haya recolectado en l. Análogamente, v pasa por al menos tantos arcos de viaje con destino en l como pedidos haya entregado en l. Se obtienen así las siguientes desigualdades válidas:

$$\begin{aligned}
&\sum_{\substack{e\in E^{\text{VIAJ}}:\\ \text{ori}(e)=l}} X_{ve} \geq \sum_{\substack{p\in P:\\ l_p^{\text{R}}=l}} \sum_{e\in E^{\text{R},p}} X_{ve} && \forall l \in L,\ v \in V, \\
&\sum_{\substack{e\in E^{\text{VIAJ}}:\\ \text{des}(e)=l}} X_{ve} \geq \sum_{\substack{p\in P:\\ l_p^{\text{E}}=l}} \sum_{e\in E^{\text{E},p}} X_{ve} && \forall l \in L,\ v \in V,
\end{aligned} \tag{RE$_1$}$$

que tienen sentido únicamente cuando en la localidad l existe algún pedido que recolectar o entregar.

Una debilidad de las desigualdades anteriores es que no tienen en cuenta los instantes de tiempo de los viajes que transportan los pedidos, que claramente deben

ser posteriores a las recolecciones y anteriores a las entregas. Para incorporar esto, se definen los conjuntos $\mathcal{I}_l^{\mathrm{R}}$ y $\mathcal{I}_l^{\mathrm{E}}$ de instantes de tiempo en los cuales finaliza un arco de recolección y respectivamente inicia un arco de entrega en l, es decir,

$$\mathcal{I}_l^{\mathrm{R}} = \{\mathrm{fin}(e) : p \in P,\ l_p^{\mathrm{R}} = l,\ e \in E^{\mathrm{R},p}\} \text{ y } \mathcal{I}_l^{\mathrm{E}} = \{\mathrm{ini}(e) : p \in P,\ l_p^{\mathrm{E}} = l,\ e \in E^{\mathrm{E},p}\}.$$

Luego son obtenidas las siguientes desigualdades válidas:

$$\begin{aligned}
\sum_{\substack{e \in E^{\mathrm{VIAJ}}: \\ \mathrm{ori}(e)=l, \\ \mathrm{ini}(e) \geq i}} X_{ve} &\geq \sum_{\substack{p \in P: \\ l_p^{\mathrm{R}} = l}} \sum_{\substack{e \in E^{\mathrm{R},p}: \\ \mathrm{fin}(e) \geq i}} X_{ve} && \forall l \in L,\ i \in \mathcal{I}_l^{\mathrm{R}},\ v \in V, \\
\sum_{\substack{e \in E^{\mathrm{VIAJ}}: \\ \mathrm{des}(e)=l, \\ \mathrm{fin}(e) \leq i}} X_{ve} &\geq \sum_{\substack{p \in P: \\ l_p^{\mathrm{E}} = l}} \sum_{\substack{e \in E^{\mathrm{E},p}: \\ \mathrm{ini}(e) \leq i}} X_{ve} && \forall l \in L,\ i \in \mathcal{I}_l^{\mathrm{E}},\ v \in V,
\end{aligned} \tag{RE$_2$}$$

que incluyen a las restricciones RE$_1$ cuando i es igual al primer instante de tiempo de $\mathcal{I}_l^{\mathrm{R}}$ y al último de $\mathcal{I}_l^{\mathrm{E}}$.

Otra alternativa para cortar tales soluciones fraccionarias surge del siguiente razonamiento. Sean una solución entera, un pedido p y un camión v. Por la construcción del digrafo y las restricciones del modelo, se puede probar que si v pasa por un arco e_1 de recolección de p y luego por un arco e_2 de entrega de p, entonces en el medio v pasa por al menos un arco de viaje con origen en l_p^{R} y por al menos un arco de viaje con destino en l_p^{E}. Lo que lleva a las siguientes desigualdades válidas:

$$\begin{aligned}
X_{ve_1} + X_{ve_2} - 1 \leq \sum_{\substack{e \in E^{\mathrm{VIAJ}}: \\ \mathrm{ori}(e)=l_p^{\mathrm{R}}, \\ \mathrm{fin}(e_1) \leq \mathrm{ini}(e), \\ \mathrm{fin}(e) \leq \mathrm{ini}(e_2)}} X_{ve} \quad \forall\ p \in P,\ e_1 \in E^{\mathrm{R},p},\ e_2 \in E^{\mathrm{E},p},\ v \in V, \\
X_{ve_1} + X_{ve_2} - 1 \leq \sum_{\substack{e \in E^{\mathrm{VIAJ}}: \\ \mathrm{des}(e)=l_p^{\mathrm{E}}, \\ \mathrm{fin}(e_1) \leq \mathrm{ini}(e), \\ \mathrm{fin}(e) \leq \mathrm{ini}(e_2)}} X_{ve} \quad \forall\ p \in P,\ e_1 \in E^{\mathrm{R},p},\ e_2 \in E^{\mathrm{E},p},\ v \in V,
\end{aligned}$$

las cuales tienen sentido únicamente cuando $\mathrm{fin}(e_1) + \mathrm{tVC}(l_p^{\mathrm{R}}, l_p^{\mathrm{E}}) \leq \mathrm{ini}(e_2)$. Desafortunadamente, estas restricciones resultan bastante débiles. Por ejemplo, no son capaces de cortar a la solución fraccionaria de la Figura 7.2.1, pues, para todo arco e de recolección o entrega, $X_{v_1 e} \leq 0,5$.

Teniendo en cuenta que, para toda solución entera, la recolección y la entrega de p se realizan exactamente una vez, es posible reforzarlas sumando sobre todos los arcos de recolección de p posteriores a e_1 y sobre todos los arcos de entrega de p anteriores a e_2, como se muestra a continuación:

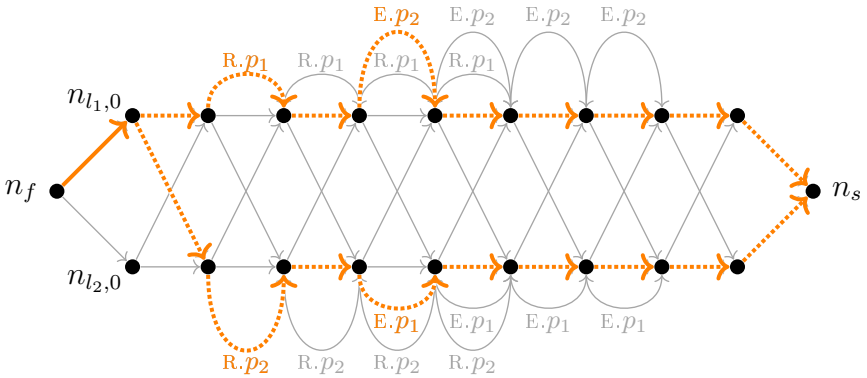
$$\sum_{\substack{e\in E^{\mathrm{R},p}:\\ \mathrm{ini}(e_1)\leq\mathrm{ini}(e)}} X_{ve} \;+\; \sum_{\substack{e\in E^{\mathrm{E},p}:\\ \mathrm{fin}(e)\leq\mathrm{fin}(e_2)}} X_{ve} - 1 \leq \sum_{\substack{e\in E^{\mathrm{VIAJ}}:\\ \mathrm{ori}(e)=l_p^{\mathrm{R}},\\ \mathrm{fin}(e_1)\leq\mathrm{ini}(e),\\ \mathrm{fin}(e)\leq\mathrm{ini}(e_2)}} X_{ve} \qquad \forall\, p\in P,\ e_1\in E^{\mathrm{R},p},\, e_2\in E^{\mathrm{E},p},\ v\in V$$

$$\sum_{\substack{e\in E^{\mathrm{R},p}:\\ \mathrm{ini}(e_1)\leq\mathrm{ini}(e)}} X_{ve} \;+\; \sum_{\substack{e\in E^{\mathrm{E},p}:\\ \mathrm{fin}(e)\leq\mathrm{fin}(e_2)}} X_{ve} - 1 \leq \sum_{\substack{e\in E^{\mathrm{VIAJ}}:\\ \mathrm{des}(e)=l_p^{\mathrm{E}}\\ \mathrm{fin}(e_1)\leq\mathrm{ini}(e),\\ \mathrm{fin}(e)\leq\mathrm{ini}(e_2)}} X_{ve} \qquad \forall\, p\in P,\ e_1\in E^{\mathrm{R},p},\, e_2\in E^{\mathrm{E},p},\ v\in V.$$

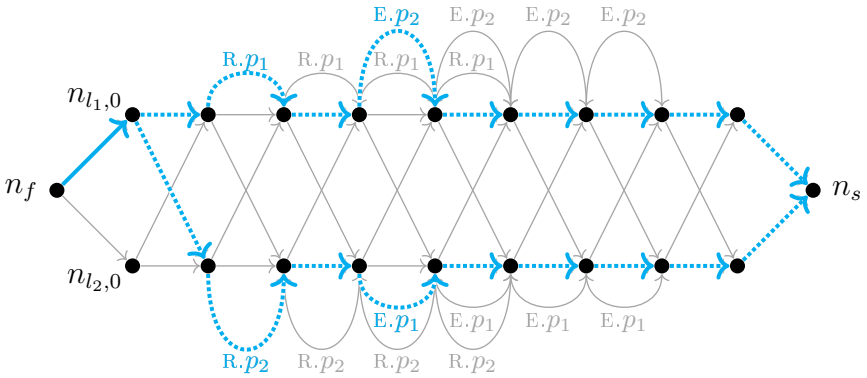
Estas restricciones ahora sí cortan a la solución fraccionaria de la Figura 7.2.1 considerando $p = p_1$, $v = v_1$ y siendo e_1 el primer arco de recolección de p_1, es decir $e_1 = (n_{l_1,1}, n_{l_1,2})$, y e_2 el último arco de entrega de p_1, es decir $e_2 = (n_{l_2,6}, n_{l_2,7})$. Por desgracia, siguen siendo débiles en instancias con más de un camión, debido a que en sus soluciones fraccionarias los pedidos suelen ser ejecutados parcialmente entre varios camiones, como muestra el siguiente ejemplo.

Ejemplo 7.2.2. Se considera la misma instancia del Ejemplo 7.2.1, salvo que ahora se dispone de un camión adicional v_2 en la localidad inicial l_1. En la Figura 7.2.2 se muestra una posible solución fraccionaria, nuevamente las variables tienen valor 1 para los arcos e de color en línea continua, valor 0,5 en los arcos de color en línea de puntos y 0 en los restantes (naranja para $X_{v_1,e}$ y cyan para $X_{v_2,e}$). Puede comprobarse fácilmente que esta solución satisface todas las restricciones de camiones del modelo, pero que no es cortada por las restricciones anteriores pues el valor del lado izquierdo es siempre menor o igual a 0. Además, su valor objetivo respecto a f_{VIAJ} es costoViajeCamion(l_1, l_2), cuando en realidad toda solución entera tiene valor objetivo acotado inferiormente por costoViajeCamion(l_1, l_2) + costoViajeCamion(l_2, l_1) (un viaje desde la localidad de recolección a la de entrega de cada pedido).

Siendo V' un subconjunto de camiones, es claro que en toda solución entera, a lo sumo un camión de V' hace la recolección y la entrega de p, luego es posible

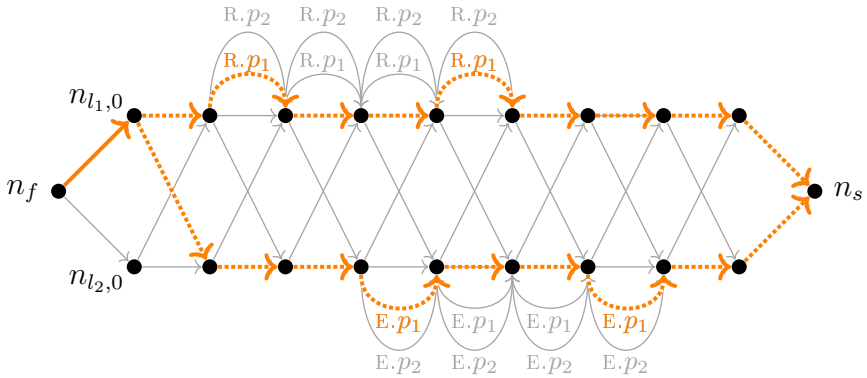


(a) Ruta del camión v_1.

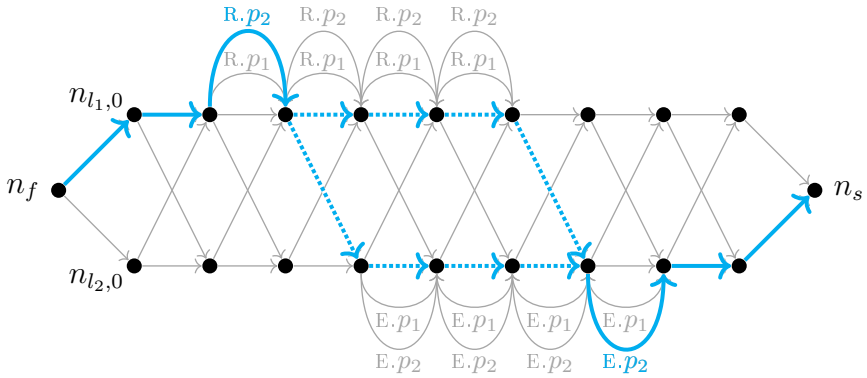


(b) Ruta del camión v_2.

Figura 7.2.2.: Solución fraccionaria en el digrafo G_{LT}.



(a) Ruta del camión v_1.



(b) Ruta del camión v_2.

Figura 7.2.3.: Solución fraccionaria en el digrafo G_{LT}.

reforzarlas de la siguiente forma:

$$\sum_{\substack{e\in E^{\mathrm{R},p}:\\ \mathrm{ini}(e_1)\leq\mathrm{ini}(e)}}\sum_{v\in V'} X_{ve} + \sum_{\substack{e\in E^{\mathrm{E},p}:\\ \mathrm{fin}(e)\leq\mathrm{fin}(e_2)}}\sum_{v\in V'} X_{ve} - 1 \leq \sum_{\substack{e\in E^{\mathrm{VIAJ}}:\\ \mathrm{ori}(e)=l_p^{\mathrm{R}},\\ \mathrm{fin}(e_1)\leq\mathrm{ini}(e),\\ \mathrm{fin}(e)\leq\mathrm{ini}(e_2)}}\sum_{v\in V'} X_{ve}$$

$$\sum_{\substack{e\in E^{\mathrm{R},p}:\\ \mathrm{ini}(e_1)\leq\mathrm{ini}(e)}}\sum_{v\in V'} X_{ve} + \sum_{\substack{e\in E^{\mathrm{E},p}:\\ \mathrm{fin}(e)\leq\mathrm{fin}(e_2)}}\sum_{v\in V'} X_{ve} - 1 \leq \sum_{\substack{e\in E^{\mathrm{VIAJ}}:\\ \mathrm{des}(e)=l_p^{\mathrm{E}},\\ \mathrm{fin}(e_1)\leq\mathrm{ini}(e),\\ \mathrm{fin}(e)\leq\mathrm{ini}(e_2)}}\sum_{v\in V'} X_{ve},$$

$$\forall\, p\in P,\ e_1\in E^{\mathrm{R},p},\ e_2\in E^{\mathrm{E},p},\ V'\subseteq V. \qquad (\mathrm{RE}_3)$$

Estas restricciones ahora sí cortan a la solución fraccionaria de la Figura 7.2.2 considerando $p = p_1$, $V' = \{v_1, v_2\}$ y siendo e_1 el primer arco de recolección de p_1, es decir $e_1 = (n_{l_1,1}, n_{l_1,2})$, y e_2 el primer arco de entrega de p_1, es decir $e_2 = (n_{l_2,3}, n_{l_2,4})$.

Por lo tanto, de lo anterior se desprende que existen soluciones fraccionarias que no pueden ser cortadas por las desigualdades RE_3 cuando V' tiene un único camión, pero si cuando se considera un subconjunto con más camiones. Sin embargo, es interesante notar que también puede ocurrir lo contrario, como muestra el siguiente ejemplo.

Ejemplo 7.2.3. Se considera la misma instancia del Ejemplo 7.2.2, salvo que el pedido p_2 tiene ahora $l_{p_2}^{\mathrm{R}} = l_1$ y $l_{p_2}^{\mathrm{E}} = l_2$. En la Figura 7.2.3 se muestra una posible solución fraccionaria (siguiendo el formato del ejemplo anterior). Puede comprobarse fácilmente que esta solución satisface todas las restricciones de camiones del modelo y también se puede probar que se verifican todas las desigualdades RE_3 con $V' = \{v_1, v_2\}$ (básicamente porque v_2 realiza el viaje de la localidad de recolección a la de entrega que debería hacer v_1 en el transporte de p_1). Sin embargo, esta solución se puede cortar con las desigualdades RE_3 tomando $p = p_1$, $V' = \{v_1\}$ y siendo e_1 el primer arco de recolección de p_1, es decir $e_1 = (n_{l_1,1}, n_{l_1,2})$, y e_2 el último arco de entrega de p_1, es decir $e_2 = (n_{l_2,6}, n_{l_2,7})$. Además, su valor objetivo respecto a f_{VIAJ} es $1,5 * \mathrm{costoViajeCamion}(l_1, l_2)$, cuando en realidad toda solución entera tiene valor objetivo acotado inferiormente por $2 * \mathrm{costoViajeCamion}(l_1, l_2)$ (un viaje desde la localidad de recolección a la de entrega de cada pedido).

Por último, surge un resultado muy importante en la adaptación de las restricciones RE_1, RE_2 y RE_3 al modelo con el digrafo G_{LTC}. Es posible reforzarlas notando que los viajes involucrados se hacen necesariamente con el camión cargado. Esto implica que cada aparición del conjunto $E^{\mathrm{VIAJ}}_{\mathrm{LTC}}$ puede reemplazarse por el subconjunto más pequeño de arcos de viaje de la parte con carga del digrafo, es decir, entre nodos con superíndice $\bullet$.

7.2.2. Restricciones de secuenciación de pedidos

Los requerimientos del problema establecen que cada camión tiene capacidad para transportar un único pedido a la vez, lo cual implica que se ejecutan secuencialmente uno después del otro. El horario en que un pedido se puede ejecutar depende del orden en que se hayan ejecutado los anteriores en esa ruta, dependiendo principalmente de las ventanas de tiempo y los tiempos de viaje entre las localidades de entrega y recolección de pedidos consecutivos. A pesar de esto, existen plazos mínimos, en cualquier solución entera, para que un camión pueda completar un subconjunto P' de pedidos, y calcularlos involucra determinar la duración mínima entre todas las posibles permutaciones de P'. En las soluciones fraccionarias, estos plazos no siempre se respetan, como se ve a continuación.

Ejemplo 7.2.4. Se considera una instancia del problema que involucra dos localidades l_1 y l_2 y un horizonte de planificación de 2 días discretizado cada 3 horas. El tiempo de viaje entre ambas localidades es de 3 horas (1 instante de tiempo). Se dispone de un único camión v en la localidad inicial l_1 y de suficientes conductores para cumplir con dos pedidos p_1 y p_2, definidos con los siguientes atributos.

Pedido p_1:

$$dia^{\mathrm{R}}_{p_1} = dia^{\mathrm{E}}_{p_1} = 0$$
$$[a^{\mathrm{R}}_{p_1}, b^{\mathrm{R}}_{p_1}] = [0, 4] \text{ (0:00-12:00)}$$
$$[a^{\mathrm{E}}_{p_1}, b^{\mathrm{E}}_{p_1}] = [2, 6] \text{ (6:00-18:00)}$$
$$l^{\mathrm{R}}_{p_1} = l_1$$
$$l^{\mathrm{E}}_{p_1} = l_2$$
$$s^{\mathrm{R}}_{p_1} = s^{\mathrm{E}}_{p_1} = 1 \text{ (3 h)}$$

Pedido p_2:

$$dia^{\mathrm{R}}_{p_2} = dia^{\mathrm{E}}_{p_2} = 0$$
$$[a^{\mathrm{R}}_{p_2}, b^{\mathrm{R}}_{p_2}] = [2, 3] \text{ (6:00-9:00)}$$
$$[a^{\mathrm{E}}_{p_2}, b^{\mathrm{E}}_{p_2}] = [4, 5] \text{ (12:00-15:00)}$$
$$l^{\mathrm{R}}_{p_2} = l_1$$
$$l^{\mathrm{E}}_{p_2} = l_2$$
$$s^{\mathrm{R}}_{p_2} = s^{\mathrm{E}}_{p_2} = 1 \text{ (3 h)}$$

En esta instancia, las siguientes dos rutas para v constituyen las de menor duración entre todas las posibles rutas que ejecutan p_1 y p_2 comenzando en el instante de tiempo 0.

Primero p_1 y luego p_2:

0-1	Recolecta p_1.
1-2	Viaja de l_1 a l_2.
2-3	Entrega p_1.
3-4	Viaja de l_2 a l_1.
4-10	Espera para recolectar p_2.
10-11	Recolecta p_2.
11-12	Viaja de l_1 a l_2.
12-13	Entrega p_2.

Primero p_2 y luego p_1:

0-2	Espera para recolectar p_2.
2-3	Recolecta p_2.
3-4	Viaja de l_1 a l_2.
4-5	Entrega p_2.
5-6	Viaja de l_2 a l_1.
6-8	Espera para recolectar p_1.
8-9	Recolecta p_1.
9-10	Viaja de l_1 a l_2.
10-11	Entrega p_1.

En consecuencia, en toda solución entera, a lo sumo un pedido entre p_1 y p_2 es recolectado en el primer día del horizonte de planificación (en el intervalo de tiempo que se extiende del instante 0 al 8). Por lo tanto, uno de los pedidos debe ser entregado con al menos un día de demora, quedando la función objetivo f_{ENTR} acotada inferiormente por:

$$\text{mín}(w_{p_1}, w_{p_2}).$$

En la Figura 7.2.4, se muestra una posible solución fraccionaria para la ruta de v en el modelo con el digrafo G_{LT}, donde la variable X_{ve} tiene valor 1 para los arcos e en línea continua naranja, valor 0,5 para los arcos en línea de puntos naranja y valor 0 para los restantes. Puede comprobarse fácilmente que esta solución satisface todas las restricciones de camiones del modelo (incluso verifica las desigualdades RE-I, RE-II y RE-III), pero que el flujo por los arcos de recolección suma 1,5 en el primer día del horizonte de planificación, excediendo el límite de 1. Además, su valor objetivo respecto a f_{ENTR} es:

$$0,5 * w_{p_2}.$$

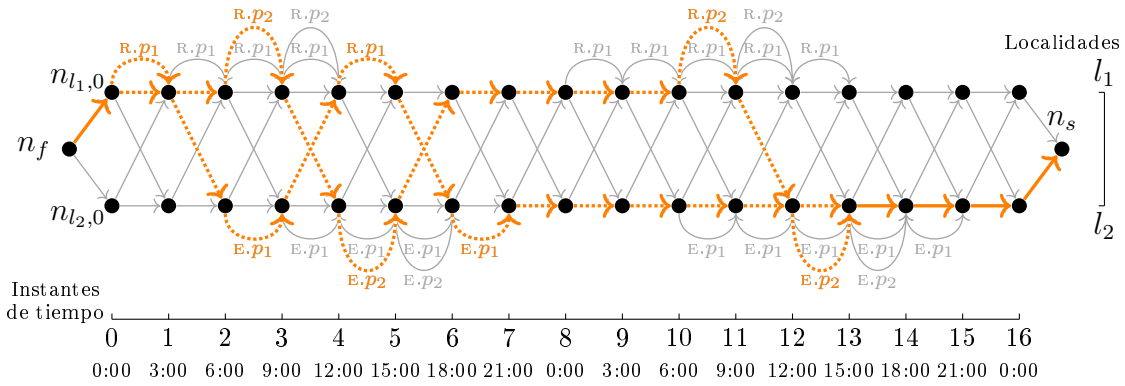


Figura 7.2.4.: Ruta de camión fraccionaria en el digrafo G_{LT}.

En las formulaciones, el valor óptimo entero puede distar notoriamente del valor óptimo de la relajación lineal debido a la existencia de soluciones fraccionarias como las de este ejemplo. El mismo inconveniente se presenta en los modelos con los digrafos G_{LTC} y G_{LTP}, donde es posible construir soluciones fraccionarias del mismo tipo. A continuación, se proponen desigualdades válidas para cortarlas.

Sean un subconjunto de pedidos P' con $|P'| \geq 2$ y un camión v. Se denota con $\text{sdur}_0^{\text{E}}(P', v)$ a la menor duración posible de una ruta para v que ejecuta en secuencia todos los pedidos de P' salvo uno y termina en el primer instante de tiempo en el que puede comenzar la entrega del último pedido. Observar que el cálculo de este parámetro es en sí mismo otro problema de optimización difícil (una variante de un PDPTW).

Determinar el valor de este parámetro por fuerza bruta involucra calcular la duración mínima de ruta para toda permutación de P'. Por ejemplo, si $|P'| = 2$, entonces $\text{sdur}_0^{\text{E}}(P', v)$ es la menor duración en que pueden ejecutarse en secuencia las siguientes tareas:

1. Viajar desde la localidad inicial de v a la localidad de recolección del primer pedido, digamos p_1.
2. Esperar a que abra la ventana de tiempo de recolección de p_1.
3. Recolectar p_1.
4. Viajar a la localidad de entrega de p_1.
5. Esperar a que abra la ventana de tiempo de entrega de p_1.
6. Entregar p_1.
7. Viajar a la localidad de recolección del segundo pedido, digamos p_2.

8. Esperar a que abra la ventana de tiempo de recolección de p_2.
9. Recolectar p_2.
10. Viajar a la localidad de entrega de p_2.
11. Esperar a que abra la ventana de tiempo de entrega de p_2.

En toda solución entera, es claro que a lo sumo $|P'| - 1$ pedidos de P' pueden ser entregados por v hasta el instante de tiempo $\text{sdur}_0^{\text{E}}(P', v)$. Esto da lugar a las siguientes desigualdades válidas:

$$\sum_{p \in P'} \sum_{\substack{e \in E^{\text{E},p} \\ \text{fin}(e) \leq \\ \text{sdur}_0^{\text{E}}(P',v)}} X_{ve} \leq |P'| - 1 \qquad \forall\ P' \subseteq P,\ v \in V, \qquad (\text{SEC}_1^{\text{E}})$$

que tienen sentido siempre que $|P'| \geq 2$ y, para cada $p \in P'$, exista al menos un arco e de entrega de p tal que $\text{fin}(e) \leq \text{sdur}_0^{\text{E}}(P', v)$.

Siguiendo un razonamiento análogo sobre los arcos de recolección, en lugar de entrega, se obtienen variaciones de estas desigualdades válidas. En este caso, se denota con $\text{sdur}_0^{\text{R}}(P', v)$ a la menor duración posible de una ruta para v que ejecuta en secuencia todos los pedidos de P' salvo uno y termina en el primer instante de tiempo en el que puede comenzar la recolección del último pedido. El cómputo de este nuevo parámetro es similar, pero ignora los puntos 9-11 de la enumeración anterior.

En toda solución entera, a lo sumo $|P'| - 1$ pedidos de P' pueden ser recolectados por v hasta el instante de tiempo $\text{sdur}_0^{\text{R}}(P', v)$, es decir:

$$\sum_{p \in P'} \sum_{\substack{e \in E^{\text{R},p} \\ \text{fin}(e) \leq \\ \text{sdur}_0^{\text{R}}(P',v)}} X_{ve} \leq |P'| - 1 \qquad \forall\ P' \subseteq P,\ v \in V, \qquad (\text{SEC}_1^{\text{R}})$$

que tienen sentido siempre que $|P'| \geq 2$ y, para todo $p \in P'$, exista al menos un arco e de recolección de p tal que $\text{fin}(e) \leq \text{sdur}_0^{\text{R}}(P', v)$.

Este enfoque supone que los pedidos de P' comienzan a ejecutarse desde el instante inicial, pero con algunas modificaciones se puede extender a otros. En este sentido, sea un instante de tiempo i arbitrario, se denota con:

- $\text{sdur}^{\text{E}}(P', i)$ a la menor duración posible de una ruta de camión que, comenzando en i, realiza la entrega de un primer pedido de P', luego ejecuta en secuencia la recolección y entrega de los demás pedidos de P' salvo uno (si hubiese) y termina en el primer instante de tiempo en el que puede comenzar la entrega del último pedido (puntos 5–11 de la enumeración anterior).

- $\text{sdur}^{\text{R}}(P', i)$ a la menor duración posible de una ruta de camión que, comenzando en i, ejecuta en secuencia la recolección y entrega de todos los pedidos de P' salvo uno y termina en el primer instante de tiempo en el que puede comenzar la recolección del último pedido (puntos 2–8 de la enumeración anterior).

Estos nuevos parámetros no dependen del camión, dado que a priori no es posible predecir su ubicación en el instante i y, por lo tanto, saber de antemano si se requiere de un viaje a la localidad de recolección/entrega del primer pedido (es decir, se ignoran los puntos 1 y 4, respectivamente).

En toda solución entera, es fácil ver que a lo sumo $|P'|-1$ pedidos de P' pueden ser entregados (respectivamente, recolectados) por un mismo camión en el intervalo que comienza en i y finaliza en $i + \text{sdur}^{\text{E}}(P', i)$ (respectivamente, finaliza en $i + \text{sdur}^{\text{R}}(P', i)$). Es decir:

$$\sum_{p \in P'} \sum_{\substack{e \in E^{\text{E},p} \\ \text{ini}(e) \geq i, \\ \text{fin}(e) \leq i + \text{sdur}^{\text{E}}(P', i)}} X_{ve} \leq |P'| - 1 \qquad \forall\ P' \subseteq P,\ i \in \mathcal{I},\ v \in V, \qquad (\text{SEC}_2^{\text{E}})$$

$$\sum_{p \in P'} \sum_{\substack{e \in E^{\text{R},p} \\ \text{ini}(e) \geq i, \\ \text{fin}(e) \leq i + \text{sdur}^{\text{R}}(P', i)}} X_{ve} \leq |P'| - 1 \qquad \forall\ P' \subseteq P,\ i \in \mathcal{I},\ v \in V, \qquad (\text{SEC}_2^{\text{R}})$$

que tienen sentido siempre que $|P'| \geq 2$, no se excedan los límites del horizonte de planificación y el rango de la segunda sumatoria no sea vacío, es decir, que todo pedido de P' tenga al menos un arco de entrega (respectivamente, recolección) en ese intervalo de tiempo.

Las restricciones SEC_1^{E}, SEC_1^{R}, SEC_2^{E} y SEC_2^{R} se pueden adaptar de manera directa a los modelos que usan el digrafo G_{LTC}. Sin embargo, sólo es posible traducirlas a los que usan G_{LTP} en instancias donde la flota tiene un único camión, debido a

que las variables X ya no están indexadas por camión y no es posible identificar qué camión cumple con cada pedido.

7.2.3. Restricciones de viaje en taxi

De la misma forma que determinados viajes que en camión están obligados a ocurrir en cualquier solución entera, también es posible identificar situaciones en donde los conductores están forzados a realizar ciertos viajes en taxi. En particular, si un camión v se encuentra inicialmente en una localidad l_v sin conductores y es utilizado para cumplir un pedido, entonces debe ocurrir al menos una de las siguientes afirmaciones: (i) otro camión lleva a un conductor hacia l_v o (ii) un conductor viaja en taxi hacia l_v. Este último viaje es necesario para que un conductor pueda llegar a l_v y ocuparse de conducir a v hacia la localidad de recolección (si es diferente de l_v) o de cargar el pedido en v (si se recolecta en l_v). Sin embargo, esta propiedad no siempre se cumple en las soluciones fraccionarias, como se muestra en el siguiente ejemplo.

Ejemplo 7.2.5. Se considera una instancia del problema que involucra dos localidades l_1 y l_2 y un horizonte de planificación de 1 día discretizado cada 2 horas. El tiempo de viaje entre ambas localidades es de 2 horas (1 instante de tiempo). Se dispone de un único camión v en la localidad inicial l_1 y un conductor c en la localidad inicial l_2 para cumplir con un pedido p_1 con los siguientes atributos.

$$\begin{aligned} dia^{\mathrm{R}}_{p_1} &= dia^{\mathrm{E}}_{p_1} = 0, \\ [a^{\mathrm{R}}_{p_1}, b^{\mathrm{R}}_{p_1}] &= [5, 6] \text{ (10:00-12:00)}, \\ [a^{\mathrm{E}}_{p_1}, b^{\mathrm{E}}_{p_1}] &= [7, 8] \text{ (14:00-16:00)}, \\ l^{\mathrm{R}}_{p_1} &= l_1, \\ l^{\mathrm{E}}_{p_1} &= l_2, \\ s^{\mathrm{R}}_{p_1} &= s^{\mathrm{E}}_{p_1} = 1 \text{ (2 h)}. \end{aligned}$$

Dado que inicialmente c y v se encuentran en localidades diferentes y no existen otros conductores ni camiones en la instancia, en cualquier solución entera c está obligado a viajar en taxi hasta l_1 para poder comenzar con la

recolección de p_1. Esto implica que el flujo de taxi por los arcos de viaje con origen en l_2 y destino l_1 debe ser mayor o igual a 1, quedando la función objetivo f_{TAXI} acotada inferiormente por:

$$\text{costoViajeTaxi}(l_1, l_2).$$

En la Figura 7.2.5, se muestra una posible solución fraccionaria para la ruta de v (en color naranja) y de c (en color magenta) en el modelo con el digrafo G_{LT}. Las variables X_{ve} e Y_{ce} tienen valor: 1 para los arcos e coloreados en línea continua, $0,5$ para los arcos e coloreados en línea de puntos y 0 para los restantes. Además, las variables Z_e tienen valor: $0,5$ para el arco de viaje e que conecta el nodo $n_{l_2,2}$ con $n_{l_1,3}$ y 0 para los arcos de viaje restantes. Puede comprobarse fácilmente que esta solución satisface todas las restricciones del modelo, pero que el flujo de taxi por los arcos de viaje suma solamente $0,5$. Por lo tanto, su valor objetivo respecto a f_{TAXI} es:

$$0,5 * \text{costoViajeTaxi}(l_1, l_2).$$

Es interesante observar que las variables X y Y asumen los valores $X_{ve} = 0,5$ e $Y_{ce} = 1$ para el arco e de viaje que conecta el nodo $n_{l_2,4}$ con $n_{l_1,5}$, pero ninguna restricción de sincronización es violada pues el flujo de conductores por e puede ser a lo sumo el doble del flujo de camiones por ese arco.

La existencia de soluciones fraccionarias como las de este ejemplo puede generar que el valor óptimo de la relajación lineal sea una cota inferior de mala calidad del valor óptimo entero. Además, los modelos con los digrafos G_{LTC} y G_{LTP} también presentan soluciones fraccionarias de este tipo. En esta sección, se proponen familias de desigualdades válidas para cortarlas.

Sean un camión v y un pedido p, tal que ningún conductor se encuentra inicialmente en la localidad inicial l_v de v. Por la construcción del digrafo y las restricciones del modelo, se puede probar que, en toda solución entera, si v pasa por alguno de los arcos de recolección de p, entonces otro camión o un taxi pasa por un arco de viaje con destino a l_v. Esta propiedad se traduce en las siguientes desigualdades

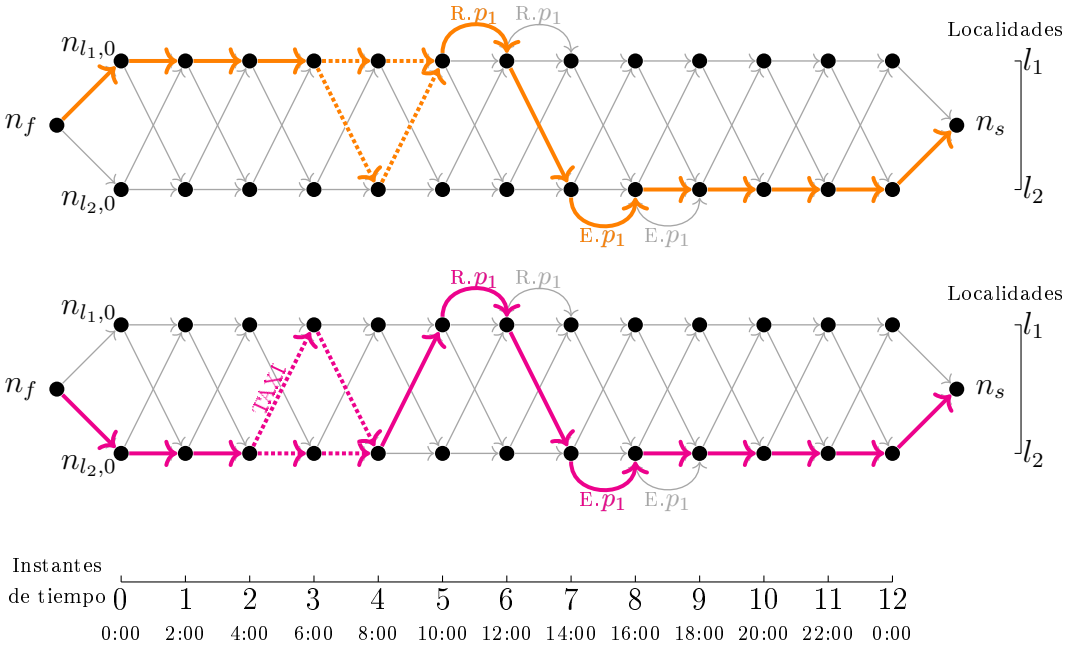


Figura 7.2.5.: Ruta de camión fraccionaria (color naranja) y ruta de conductor fraccionaria (color magenta) en el digrafo G_{LT}.

válidas:

$$\sum_{e \in E^{\text{R},p}} X_{ve} \leq \sum_{\substack{e \in E^{\text{VIAJ}}: \\ \text{des}(e)=l_v}} \sum_{\substack{v' \in V: \\ v' \neq v}} X_{v'e} + \sum_{\substack{e \in E^{\text{VIAJ}}: \\ \text{des}(e)=l_v}} Z_e \quad \forall v \in V,\ p \in P \text{ tal que } l_v \notin L_C. \quad (\text{TAXI}_1)$$

Sin embargo, estas restricciones no tienen en cuenta el orden temporal de este viaje, el cual lógicamente debe ocurrir antes de que v pase por un arco e de recolección de p. Más aún, debe ocurrir con suficiente antelación para que el camión haga tiempo a viajar desde l_v a la localidad de recolección l_p^{R} de p, siempre que ellas sean diferentes. Por lo tanto, es posible reescribirlas como sigue:

$$X_{ve} \leq \sum_{\substack{e' \in E^{\text{VIAJ}}: \\ \text{des}(e')=l_v, \\ \text{fin}(e') \leq \\ \text{ini}(e)-\text{tVC}(l_v,l_p^{\text{R}})}} \sum_{\substack{v' \in V: \\ v' \neq v}} X_{v'e'} + \sum_{\substack{e' \in E^{\text{VIAJ}}: \\ \text{des}(e')=l_v, \\ \text{fin}(e') \leq \\ \text{ini}(e)-\text{tVC}(l_v,l_p^{\text{R}})}} Z_{e'} \qquad \forall v \in V,\ p \in P, e \in E^{\text{R},p} \text{ tal que } l_v \notin L_C.$$

Observar que, para toda solución entera, si $X_{ve} = 1$, entonces las restricciones del modelo garantizan que $X_{ve'} = 0$ para todo arco e' de recolección de p anterior a e.

Por lo tanto, es posible reforzarlas como sigue:

$$\sum_{\substack{e' \in E^{\mathrm{R},p}: \\ \mathrm{ini}(e') \leq \mathrm{ini}(e)}} X_{ve} \leq \sum_{\substack{e' \in E^{\mathrm{VIAJ}}: \\ \mathrm{des}(e') = l_v, \\ \mathrm{fin}(e') \leq \\ \mathrm{ini}(e) - \mathrm{tVC}(l_v, l_p^{\mathrm{R}})}} \sum_{\substack{v' \in V: \\ v' \neq v}} X_{v'e'} + \sum_{\substack{e' \in E^{\mathrm{VIAJ}}: \\ \mathrm{des}(e') = l_v, \\ \mathrm{fin}(e') \leq \\ \mathrm{ini}(e) - \mathrm{tVC}(l_v, l_p^{\mathrm{R}})}} Z_{e'} \qquad \begin{array}{r} \forall v \in V,\ p \in P, e \in E^{\mathrm{R},p} \\ \text{tal que } l_v \notin L_C. \end{array} \qquad (\mathrm{TAXI}_2)$$

En particular, cuando la instancia posee un único camión, el primer término del lado derecho desaparece. Es decir, la única posibilidad para cumplir con el pedido es que algún conductor viaje en taxi a l_v.

Es posible adaptar estas restricciones a las demás formulaciones. Por ejemplo, si las rutas de los conductores se modelan con el digrafo G_{LTX} (y las de los camiones con G_{LT}), entonces las TAXI_2 se pueden expresar de la siguiente forma:

$$\sum_{\substack{e' \in E_{\mathrm{LT}}^{\mathrm{R},p}: \\ \mathrm{ini}(e') \leq \mathrm{ini}(e)}} X_{ve} \leq \sum_{\substack{e' \in E_{\mathrm{LT}}^{\mathrm{VIAJ}}: \\ \mathrm{des}(e') = l_v, \\ \mathrm{fin}(e') \leq \\ \mathrm{ini}(e) - \mathrm{tVC}(l_v, l_p^{\mathrm{R}})}} \sum_{\substack{v' \in V: \\ v' \neq v}} X_{v'e'} + \sum_{\substack{e' \in E_{\mathrm{LTX}}^{\mathrm{TAXI}}: \\ \mathrm{des}(e') = l_v, \\ \mathrm{fin}(e') \leq \\ \mathrm{ini}(e) - \mathrm{tVC}(l_v, l_p^{\mathrm{R}})}} \sum_{c \in C} Y_{ce'} \qquad \begin{array}{r} \forall v \in V,\ p \in P, e \in E_{\mathrm{LT}}^{\mathrm{R},p} \\ \text{tal que } l_v \notin L_C. \end{array}$$

También es posible adaptarlas cuando las rutas de camiones se modelan con el digrafo G_{LTC}, aunque las mismas se omiten para simplificar la redacción debido a sus similitudes. Sin embargo, cuando ellas se modelan con G_{LTP}, no es posible adaptarlas si la instancia posee más de un camión, dado que las variables X dejan de estar indexadas por camión y no es posible identificar qué camión cumple con cada pedido. En el caso de instancias con un único camión, cuando las rutas de camiones se modelan con G_{LTP} y las de conductores con G_{LT}, las TAXI_2 se pueden expresar como sigue:

$$\sum_{\substack{e' \in E_{\mathrm{LTP}}^{\mathrm{R},p}: \\ \mathrm{ini}(e') \leq \mathrm{ini}(e)}} X_e \leq \sum_{\substack{e' \in E_{\mathrm{LT}}^{\mathrm{VIAJ}}: \\ \mathrm{des}(e') = l_v, \\ \mathrm{fin}(e') \leq \\ \mathrm{ini}(e) - \mathrm{tVT}(l_v, l_p^{\mathrm{R}})}} Z_{e'} \qquad \begin{array}{r} \forall v \in V,\ p \in P, e \in E_{\mathrm{LTP}}^{\mathrm{R},p} \\ \text{tal que } l_v \notin L_C. \end{array}$$

7.2.4. Otras desigualdades válidas

Por último, se mencionan otras desigualdades válidas que también han sido consideradas durante este trabajo de tesis. Es importante aclarar que todas ellas tienen

potencial de ser usadas como cortes en los algoritmos de optimización, visto que su incorporación en las relajaciones lineales elevó el valor óptimo fraccionario para algunas instancias particulares en pruebas preliminares. Sin embargo, se decide presentarlas en una sección separada debido a que los incrementos observados en los valores objetivos han sido menos marcados en comparación con las descriptas anteriormente. Más allá de esto, la búsqueda de desigualdades válidas sigue siendo un enorme desafío y conocer posibles estructuras de restricciones es sumamente importante para inspirar el diseño de desigualdades válidas más ajustadas en investigaciones futuras.

Secuenciación de pedidos para múltiples camiones

En instancias con múltiples camiones, es usual que en las soluciones óptimas fraccionarias los pedidos sean ejecutados parcialmente entre varios camiones. Por ejemplo, dados un pedido p y dos camiones v_1 y v_2, es posible que el flujo de v_1 por los arcos de recolección de p sume 0,5 y análogamente para v_2. Por este motivo, la efectividad de las restricciones de secuenciación de pedidos de la Sección 7.2.2 se reduce drásticamente en instancias con múltiples camiones y vale la pena pensar generalizaciones para subconjuntos de camiones de tamaño arbitrario.

En este sentido, sean dos camiones distintos v_1 y v_2 y un subconjunto P' de pedidos con $|P'| \geq 3$. Es evidente que en toda solución entera, si v_1 y v_2 ejecutan todos los pedidos de P', entonces uno de los camiones debe ejecutar al menos

$$\left\lceil \frac{|P'|}{2} \right\rceil$$

pedidos, mientras que los pedidos restantes pueden ser ejecutados por el mismo camión o en paralelo por el otro camión. Sea $[i_1, i_2]$ un intervalo de tiempo que comienza en un instante de tiempo i_1 y finaliza en

$$i_2 = i_1 + \text{mín} \left\{ \text{sdur}^{\text{R}}(P'', i_1) : P'' \subseteq P' \wedge |P''| = \left\lceil \frac{|P'|}{2} \right\rceil \right\},$$

es decir, entre todos los subconjuntos P'' de P' de cardinal $\lceil |P'|/2 \rceil$, i_2 es el menor instante de tiempo en que un camión que comienza en i_1 puede ejecutar secuencialmente todos los pedidos de P'' salvo uno y quedar listo para recolectar el último

pedido. Entonces en el intervalo $[i_1, i_2]$, el flujo de v_1 y v_2 por los arcos de recolección asociados a los pedidos de P' puede sumar a lo sumo

$$2\left(\left\lceil \frac{|P'|}{2} \right\rceil - 1\right).$$

De esta forma, surgen las siguientes desigualdades válidas:

$$\sum_{p\in P'} \sum_{\substack{e\in E^{\mathrm{R},p} \\ \mathrm{ini}(e)\geq i_1, \\ \mathrm{fin}(e)\leq i_2}} (X_{v_1 e} + X_{v_2 e}) \leq 2\left(\left\lceil \frac{|P'|}{2} \right\rceil - 1\right) \quad \forall\ P' \subseteq P,\ i_1 \in \mathcal{I},\ \{v_1, v_2\} \subseteq V,$$

y también es posible generalizarlas para cualquier subconjunto V' de dos o más camiones:

$$\sum_{p\in P'} \sum_{v\in V'} \sum_{\substack{e\in E^{\mathrm{R},p} \\ \mathrm{ini}(e)\geq i_1, \\ \mathrm{fin}(e)\leq i_2}} X_{ve} \leq |V'|\left(\left\lceil \frac{|P'|}{|V'|} \right\rceil - 1\right) \qquad \forall\ P' \subseteq P,\ i_1 \in \mathcal{I},\ V' \subseteq V,$$

tal que $|P'| \geq |V'| + 1$, $|V'| \geq 2$, $i_2 = i_1 + \mathrm{mín}\{\mathrm{sdur}^{\mathrm{R}}(P'', i_1) : P'' \subseteq P' \wedge |P''| = \lceil {}^{|P'|}/_{|V'|} \rceil\}$.

Viaje en taxi para múltiples camiones

Sea una localidad l, se denota con V_l y C_l a los subconjuntos de camiones y conductores que tienen a l como localidad inicial, respectivamente. Dada una instancia que posee una localidad l con $|C_l| < |V_l|$, entonces, para toda solución entera donde cada camión de V_l viaje a una localidad diferente de l, debe existir una cantidad de conductores mayor o igual a $|V_l| - |C_l|$ que viaje en camión o en taxi hacia l (no es necesario que sean todos diferentes). De no cumplirse, algún camión no tendría conductor para realizar dicho viaje. Las restricciones de viaje en taxi de la Sección 7.2.3 pueden pensarse como un caso particular donde $V_l = \{v\}$, para algún $v \in V$, y $C_l = \emptyset$.

El principal desafío para escribir tales desigualdades válidas es lograr expresar que todos los camiones de V_l abandonan l. Lo más intuitivo es considerar el flujo por los arcos de viaje con origen en l, sin embargo esto no es efectivo en la práctica dado que en las soluciones fraccionarias el flujo de un camión por un arco de viaje

suele ser bastante inferior a 1. El enfoque aquí propuesto se basa en las siguientes observaciones. Por un lado, si un camión entrega un pedido, entonces de seguro abandona su localidad inicial antes de realizar dicha descarga, por ejemplo, para viajar de la localidad de recolección a la de entrega (que siempre son diferentes entre sí). Por el otro lado, todos los camiones de V_l son simétricos y por ende pueden intercambiarse sus rutas. Con el objetivo de mantener simple la redacción, se presentan posibles desigualdades válidas para el caso particular de una localidad l con $V_l = \{v_1, v_2\}$ y $|C_l| = 1$, aunque el desarrollo es extensible a otros casos.

Si la instancia es factible, entonces existirá una solución óptima entera donde v_2 ejecuta pedidos sólo si v_1 también y, en ese caso, el primer viaje con origen en l que v_1 realiza inicia en un instante de tiempo anterior o igual al primer viaje con origen en l que v_2 realiza. Notar que si una solución entera no verifica esto, entonces fácilmente se puede construir una que lo verifique y que preserve el valor objetivo mediante el intercambio de las rutas de v_2 y v_1. Luego, se puede decir que si v_2 entrega un pedido p, entonces v_1 también entrega algún pedido, y por lo tanto ambos camiones abandonan eventualmente su localidad inicial l, siendo necesarios dos conductores en l, uno para ejecutar el primer viaje de v_1 y otro para el primero de v_2. Observar que estos dos viajes podrían ser realizados por el mismo conductor, pero de seguro luego de ejecutar el primero debe regresar a l (en camión o en taxi) para ejecutar el otro. Dado que $|C_l| = 1$, es decir, se dispone de un único conductor en l, alguien debe viajar en camión o en taxi hacia l antes de que v_2 realice la entrega de p. Se obtienen así las siguientes desigualdades válidas:

$$\sum_{\substack{e' \in E^{\mathrm{E},p}: \\ \mathrm{ini}(e') \leq \mathrm{ini}(e)}} X_{v_2 e'} \leq \sum_{\substack{e' \in E^{\mathrm{VIAJ}}: \\ \mathrm{des}(e') = l, \\ \mathrm{fin}(e') \leq \mathrm{ini}(e)}} \sum_{v' \in V} X_{v' e'} + \sum_{\substack{e' \in E^{\mathrm{VIAJ}}: \\ \mathrm{des}(e') = l, \\ \mathrm{fin}(e') \leq \mathrm{ini}(e)}} Z_{e'} \qquad \begin{array}{r} \forall\, l \in L,\ p \in P,\ e \in E^{\mathrm{E},p} \text{ tal que} \\ V_l = \{v_1, v_2\} \text{ y } |C_l| = 1. \end{array}$$

Sincronización de conductor individual

Dado un conductor c y un arco e de viaje, es fácil ver que las restricciones de activación de variables de taxi implican a las siguientes:

$$Y_{ce} \leq 2 \sum_{v \in V} X_{ve} + Z_e.$$

Considerando que las variables X e Y son binarias, es posible ajustarlas como sigue:

$$Y_{ce} \leq \sum_{v \in V} X_{ve} + Z_e.$$

Como es esperable, todas las soluciones enteras verifican estás últimas desigualdades, pero esto en general no ocurre en las soluciones fraccionarias. Por ejemplo, en una instancia con dos conductores c_1 y c_2 y un único camión v, las variables podrían asumir los valores: $Y_{c_1e} = 1$, $Y_{c_2e} = 0$, $X_{ve} = 0,5$ y $Z_e = 0$.

A partir de las restricciones de sincronización, es posible definir desigualdades válidas similares para los arcos de recolección y de entrega.

Flujo ajustado de camiones

Para todo camión $v \in V$, nodo $n \in N_{\text{LTC}} \setminus \{n_f, n_s\}$ y arcos $e_1 \in \Gamma^-(n)$ y $e_2 \in \Gamma^+(n)$, es fácil ver que las restricciones de flujo de camiones implican a las siguientes:

$$X_{ve_1} \leq \sum_{e \in \Gamma^+(n)} X_{ve} \qquad \text{y} \qquad \sum_{e \in \Gamma^-(n)} X_{ve} \geq X_{ve_2}.$$

Dado que e_1 y e_2 son adyacentes, la estructura de G_{LTC} garantiza que se da uno de los siguientes casos: (i) alguno de ellos no es de recolección o de entrega o (ii) uno es de entrega y el otro de recolección pero asociados a pedidos diferentes. Ambos arcos no pueden ser de recolección (ni ambos de entrega) debido a que este tipo de arco conecta nodos de superíndice $\circ$ con nodos de superíndice $\bullet$ (respectivamente, nodos de superíndice $\bullet$ con nodos de superíndice $\circ$). A su vez, tampoco puede suceder que uno sea de entrega y el otro de recolección pero asociados al mismo pedido, debido a que las localidades de recolección y de entrega de un mismo pedido son siempre diferentes.

Para toda solución entera, es claro que si un camión pasa por e_1 y por e_2 estando en el caso (ii), entonces necesariamente e_1 debe ser de entrega y e_2 de recolección. De lo contrario, es decir, si e_1 fuese de recolección y e_2 de entrega, entonces el camión entregaría un pedido diferente al que acaba de recolectar. Luego, es posible ajustar

las desigualdades válidas anteriores de la siguiente forma:

$$X_{ve_1} \leq \sum_{e \in \Gamma^+(n) \setminus E^{\mathrm{E}}_{\mathrm{LTC}}} X_{ve} \qquad \forall\, v \in V,\ n \in N_{\mathrm{LTC}} \setminus \{n_f, n_s\},\ e_1 \in \Gamma^-(n) \cap E^{\mathrm{R}}_{\mathrm{LTC}},$$

$$\sum_{e \in \Gamma^-(n) \setminus E^{\mathrm{R}}_{\mathrm{LTC}}} X_{ve} \geq X_{ve_2} \qquad \forall\, v \in V,\ n \in N_{\mathrm{LTC}} \setminus \{n_f, n_s\},\ e_2 \in \Gamma^+(n) \cap E^{\mathrm{E}}_{\mathrm{LTC}}.$$

8. Experiencias computacionales

Este capítulo se enfoca en el desarrollo y configuración de un algoritmo de optimización para la resolución exacta del problema en estudio. El algoritmo desarrollado hace uso del solver CPLEX [IBM, 2021] para la optimización de las formulaciones de PLE e incorpora los desarrollos teóricos presentados en los capítulos anteriores, tales como una heurística inicial para podar nodos desde el arranque y desigualdades válidas para reforzar las relajaciones lineales. Estas incorporaciones logran mejorar ampliamente los resultados obtenidos respecto a CPLEX sin ninguna funcionalidad adicional. También se analizan los límites de estos modelos y se comparan con los desarrollos realizados en la primera parte de la tesis. Cada una de las experiencias computacionales llevadas a cabo se acompaña de cuadros y gráficas para mostrar los resultados obtenidos y también se presenta una discusión de los mismos.

Para la realización de las experiencias computacionales se dispone de una computadora equipada con un procesador Intel i7 3.00GHz (usando un único thread), 7.7GB de memoria, sistema operativo Ubuntu 18.04 64bits. Todas las implementaciones fueron escritas en el lenguaje de programación `C++11` y compilado con GCC 9.3.0. La versión de CPLEX es 20.1.0 y se usa un límite de tiempo 2 horas (7200 segundos) por instancia, dejando todos los demás parámetros en su valor por defecto. Para futuras comparaciones, el programa de benchmark *dfmax* reporta un promedio de 2,86 segundos (tiempo de usuario) sobre 10 ejecuciones para la instancia r500.5.b (disponible en [DIMACS Challenges, 2002]).

8.1. Generación de instancias

Con el objetivo de medir el desempeño de los los algoritmos de optimización se utilizan instancias de prueba, generadas mediante el procedimiento detallado en la Sección 4.1. En esta ocasión, se usaron mapas más chicos para que el tamaño de las formulaciones de PLE sea tratable. Las Figuras 8.1.1a y 8.1.1b muestran los mapas utilizados, cuyas distancias y tiempos de viaje pueden extraerse de los Cuadros 4.1.1 y 4.1.2.

En particular, para este capítulo se generaron los siguientes conjuntos de prueba, cada uno compuesto por 15 instancias distintas.

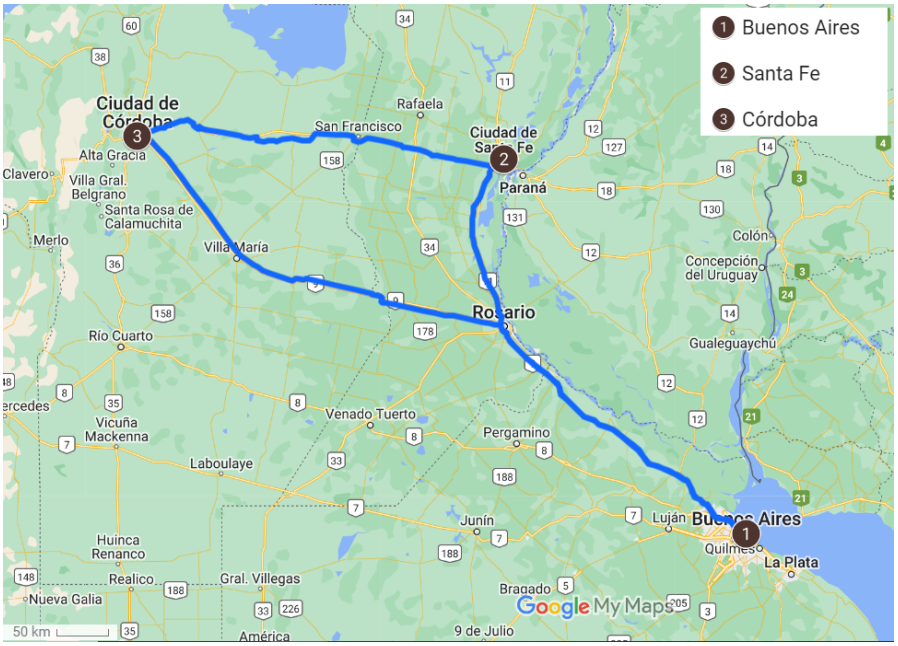
- `Set2.1` contiene las instancias más chicas. Considera el mapa de 3 ciudades y contiene 5 instancias por cada valor de $|P| \in \{4, 5, 6\}$. Los valores para los demás parámetros están fijos en $H = 7$, $|V| = 1$ y $|C| = 2$.

- `Set2.2` contiene instancias con mayor número de pedidos que `Set2.1`. Considera 5 instancias por cada valor de $|P| \in \{7, 8, 9\}$ y los mismos valores para los demás parámetros.

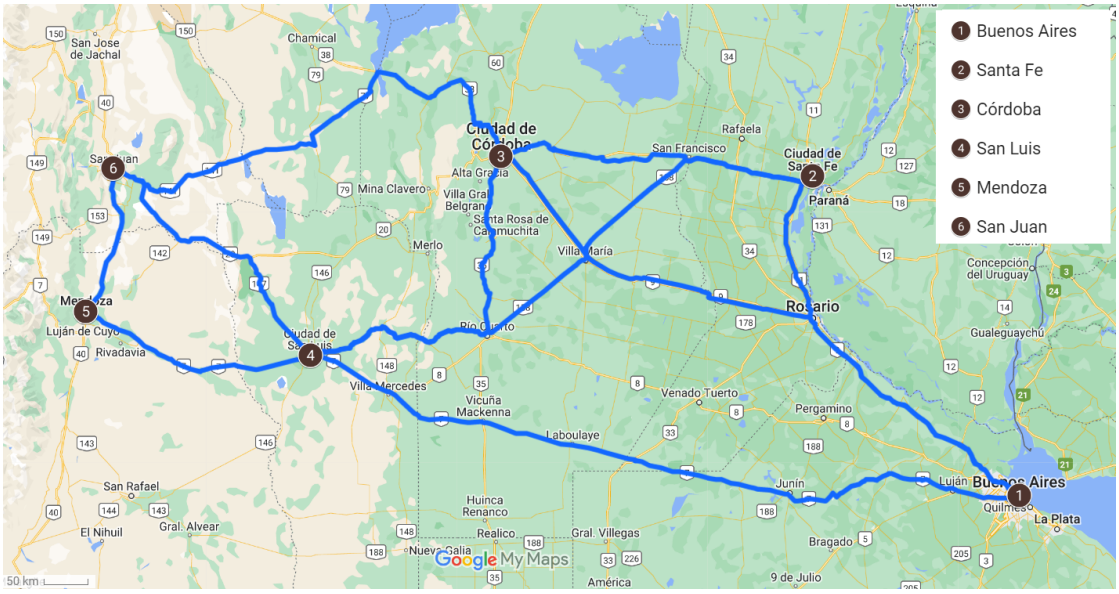
- `Set2.3` contiene instancias con mayor número de camiones y conductores. Considera 5 instancias por cada valor de $|P| \in \{8, 9, 10\}$ y los valores para los demás parámetros están fijos en $H = 7$, $|V| = 2$ y $|C| = 4$.

- `Set2.4` utiliza el mapa de 6 ciudades y contiene 5 instancias por cada valor de $|P| \in \{4, 5, 6\}$. Los valores para los demás parámetros están fijos en $H = 7$, $|V| = 1$ y $|C| = 2$.

En la generación se descartaron instancias que sean *fáciles* de resolver, específicamente aquellas donde el valor objetivo de una solución óptima de la primera etapa coincide con el valor objetivo de una solución factible del problema integrado, ya que entonces dicha solución factible es óptima y no es necesario recurrir a un algoritmo de B&C. Para computar una solución óptima de la primera etapa se optimizó la formulación E1 (de la Sección 2.1.4) con el solver CPLEX (el tamaño de las instancias permite resolverlas por optimalidad en pocos segundos), mientras que una solución factible se obtuvo con el procedimiento heurístico secuencial que se describirá en la Sección 8.4.



(a) Mapa de 3 ciudades.



(b) Mapa de 6 ciudades.

Figura 8.1.1.: Mapas con las ciudades y rutas consideradas, extraído de *Google Maps*.

En las próximas secciones se exponen las experiencias computacionales llevadas a cabo. Para usar como referencia durante la lectura, en el Cuadro 8.1.1 se resumen todos los algoritmos evaluados, de modo de poder identificar rápidamente las componentes activadas en cada uno de ellos. Cada fila de la tabla se corresponde con un algoritmo, donde la primera columna indica la formulación de PLE usada como base, la segunda el nombre del algoritmo y la tercera referencia el cuadro donde es usado. Las demás columnas representan las distintas componentes de los algoritmos: restricciones alternativas, heurística inicial y desigualdades válidas. Como regla general, las distintas componentes se incorporan incrementalmente y el nombre del algoritmo hace referencia a la última incorporación. Por ejemplo, el algoritmo RE_1 (RF) para LTC-LTX incorpora las desigualdades válidas homónimas, manteniendo para las restricciones alternativas y la heurística inicial la mejor configuración obtenida previamente.

8.2. Formulaciones

La primera experiencia computacional consiste en comparar el desempeño de las seis formulaciones de PLE presentadas en el Capítulo 6: LT-LT, LT-LTX, LTC-LT, LTC-LTX, LTP-LT y LTP-LTX.

En cada caso, se implementó un programa que lee una instancia, escribe la formulación correspondiente y la optimiza con el solver de PLE de CPLEX. Los programas fueron escritos en `C++11` y hacen uso de la API de Concert Technology para la comunicación con CPLEX. Todos los parámetros fueron usados en su valor por defecto, salvo el límite tiempo de 7200 segundos (por instancia), un único thread (sin paralelismo) y el modo de optimización determinista (múltiples ejecuciones del mismo modelo, con mismos parámetros y misma máquina, producen misma salida). En particular, se mantiene la estrategia de búsqueda que CPLEX aplica por defecto, esto es *búsqueda dinámica* en lugar de *branch-and-cut* convencional, que en general permite encontrar soluciones factibles y óptimas más rápidamente. A su vez, todos los cortes de CPLEX están activados con su comportamiento por defecto. Estos parámetros mantendrán estos valores, salvo que se explicite lo contrario.

	Algoritmo	Ref.	Restr. alternativas			Heur.	Desigualdades válidas		
			Precedencia	Descanso semanal	Sincronización		Viaje reco. entrega	Secuen. de pedidos	Viaje en taxi
LTC-LTX	Inicial	8.2.1	$PREC_1$	$DESC_1$	$SINC_1$				
	$PREC_2$	8.3.1	$PREC_2$	$DESC_1$	$SINC_1$				
	$DESC_2$		$PREC_1$	$DESC_2$	$SINC_1$				
	$SINC_2$		$PREC_1$	$DESC_1$	$SINC_2$	×	×		
	$SINC_3$		$PREC_1$	$DESC_1$	$SINC_3$				
	$PREC_2$ + $SINC_2$	8.3.2	$PREC_2$	$DESC_1$	$SINC_2$			×	
	$PREC_2$ + $SINC_3$		$PREC_2$	$DESC_1$	$SINC_3$				
	Heur. inicial	8.4.1	$PREC_2$	$DESC_1$	$SINC_2$	✓			×
	RE_1 (RF)	8.5.4					RE_1 (RF)		
	RE_2 (RF)						RE_2 (RF)		
	RE_2 (PC)						RE_2 (PC)		
	RE_3-V_2-$e_\cup$ (PC)						RE_3-V_2-$e_\cup$ (PC)		
	RE_3-V_2-$e_\cap$ (RF)						RE_3-V_2-$e_\cap$ (RF)		
	SEC_1-P_4 (RF)	8.5.5				✓		SEC_1-P_4 (RF)	
	SEC_1-P_4 (PC)							SEC_1-P_4 (PC)	
	SEC_2-P_4 (PC)							SEC_2-P_4 (PC)	
	$TAXI_1$ (RF)	8.5.7					RE_2 (RF)	×	$TAXI_1$ (RF)
	$TAXI_2$ (RF)								$TAXI_2$ (RF)
	$TAXI_2$ (PC)								$TAXI_2$ (PC)
	Final	8.6.2							$TAXI_1$ (RF)
LTP-LTX	Inicial	8.2.1	No aplica	$DESC_1$	$SINC_1$		No aplica		
	$DESC_2$	8.3.4		$DESC_2$	$SINC_1$	×		×	
	$SINC_2$			$DESC_1$	$SINC_2$				
	$SINC_3$			$DESC_1$	$SINC_3$				
	Heur. inicial	8.4.1		$DESC_1$	$SINC_2$	✓			×
	SEC_1-P_4 (RF)	8.5.6						SEC_1-P_4 (RF)	
	SEC_1-P_4 (PC)							SEC_1-P_4 (PC)	
	SEC_2-P_4 (PC)							SEC_2-P_4 (PC)	
	$TAXI_1$ (RF)	8.5.8				✓		SEC_1-P_4 (PC)	$TAXI_1$ (RF)
	$TAXI_2$ (RF)								$TAXI_2$ (RF)
	$TAXI_2$ (PC)								$TAXI_2$ (PC)
	Final	8.6.2							$TAXI_2$ (PC)

Cuadro 8.1.1.: Resumen de las componentes activadas en los algoritmos evaluados.

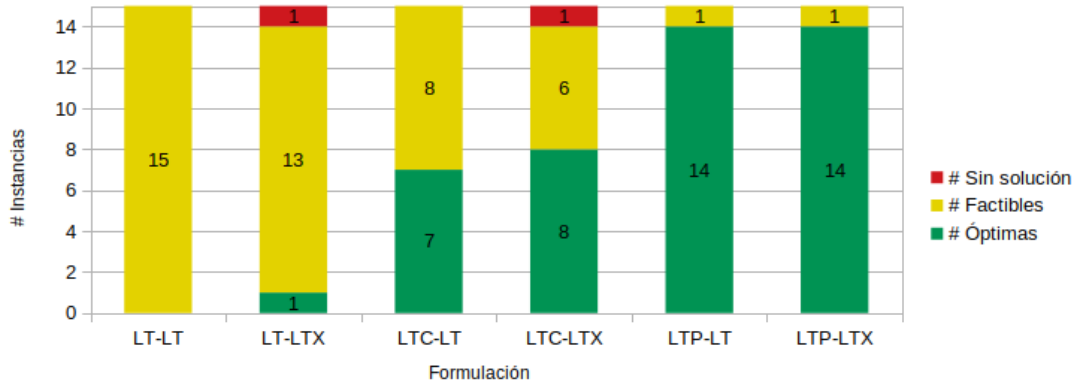
Se opta por minimizar una única función objetivo que combina f_{ENTR}, f_{VIAJ} y f_{TAXI} en una suma ponderada con pesos iguales a 1, es decir, dando la misma importancia a todas ellas. Por otro lado, no se aplica normalización por dos motivos. En primer lugar, no es posible debido a que los métodos necesarios requieren información desconocida para este problema, en particular, para cada instancia, el rango de valores posibles para cada función objetivo por separado. En segundo lugar, en las experiencias computacionales que se realizarán en esta segunda parte de la tesis, se observa que las soluciones encontradas tienen valores *parejos* respecto a cada uno de los tres objetivos por separado, es decir, ningún objetivo parece ser priorizado por sobre los demás, al menos para el tamaño de instancias consideradas (`Set2.1` a `Set2.4`).

El Cuadro 8.2.1a resume los resultados obtenidos para el `Set2.1`. La fila "Formulación" indica la formulación considerada en cada columna, "# Variables" es el promedio del número de variables, "# Restricciones" es el promedio del número de restricciones, "# Óptimas" es el número de instancias resueltas por optimalidad dentro del tiempo límite, "# Factibles" es el número de instancias para las cuales se encontró al menos una solución factible pero no se logró probar la optimalidad dentro del tiempo límite, "# Sin solución" es el número de instancias para las cuales no se encontró ninguna solución factible dentro del tiempo límite, "Tiempo (s)" es el promedio del tiempo de ejecución en segundos (para las instancias no resueltas por optimalidad se considera un tiempo de ejecución igual al límite de tiempo), "Tiempo opt. (s)" es el promedio del tiempo de ejecución en segundos para las instancias resueltas por optimalidad, "Nodos opt." es el promedio del número de nodos explorados en las instancias resueltas por optimalidad, "Gap (%)" es el promedio del porcentaje de gap reportado por CPLEX (para las instancias resueltas por optimalidad se considera un gap de 0 % y en las que no se encuentra solución factible se excluyen del promedio) y "Gap fact. (%)" es el promedio del porcentaje de gap para las instancias reportadas como factibles. Además, en la Figura 8.2.1b, se muestra un gráfico de barras con los valores de "# Óptimas", "# Factibles", y "# Sin solución" alcanzados por cada formulación.

Considerando el número de instancias resueltas por optimalidad, las formulaciones con peor desempeño son LT-LT y LT-LTX, le siguen LTC-LT y LTC-LTX y las mejores son LTP-LT y LTP-LTX, a pesar del incremento en el número de variables y restricciones. Estos resultados muestran que las formulaciones se vuelven suma-

Formulación	LT-LT	LT-LTX	LTC-LT	LTC-LTX	LTP-LT	LTP-LTX
# Variables	7702	8682	9186	10166	15122	16102
# Restricciones	5813	5813	6151	6151	7828	7828
# Óptimas	0	1	7	8	14	14
# Factibles	15	13	8	6	1	1
# Sin solución	0	1	0	1	0	0
Tiempo (s)	7200	7109	4863	4597	1185	894
Tiempo opt. (s)	-	5828	2192	2319	756	444
Nodos opt.	-	139372	13290	9154	1647	1915
Gap (%)	55	49	15	14	0,4	0,8
Gap fact. (%)	55	53	29	33	7	11

(a) Cuadro con el resumen de los resultados por formulación.



(b) Gráfico del número de instancias resueltas por formulación.

Cuadro 8.2.1.: Comparación de las formulaciones sobre `Set2.1`.

mente más competitivas con la incorporación de mayor estructura en el digrafo que modela las rutas de los camiones.

Al comparar LT-LT con LT-LTX, LTC-LT con LTC-LTX y LTP-LT con LTP-LTX, se puede apreciar un comportamiento más parejo, pero con algunas diferencias a favor de la segunda formulación en lo que respecta al número de instancias resueltas por optimalidad y el tiempo de ejecución, a pesar del mayor número de variables. Estos resultados indican que la elección del digrafo para modelar las rutas de los conductores repercute en menor medida sobre la performance, en comparación con el análisis previo.

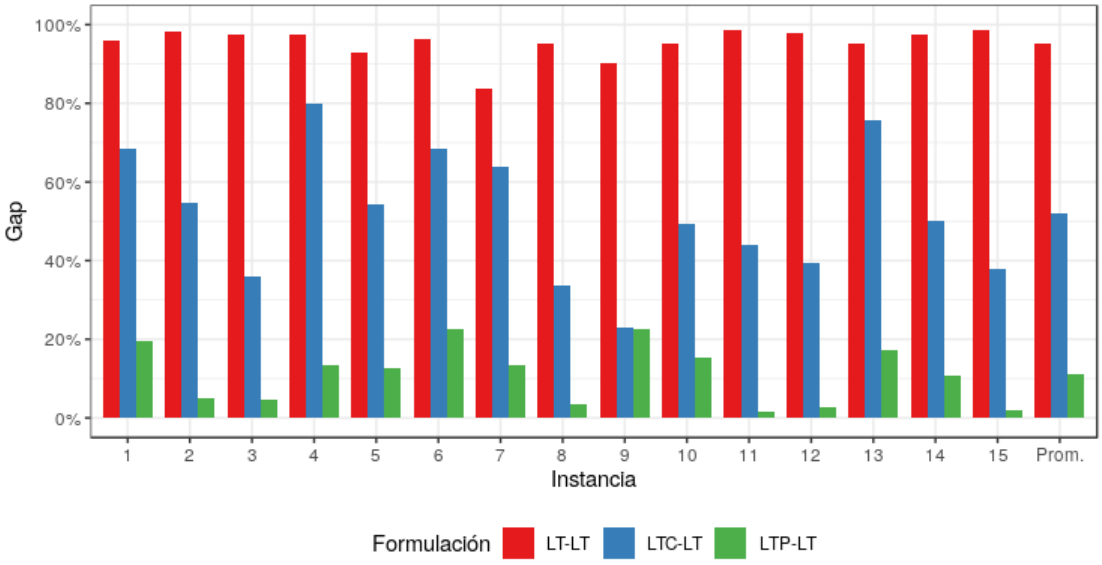
Las diferencias observadas en la performance podrían deberse al menor o mayor ajuste que tienen las relajaciones lineales de cada una de las formulaciones. Para explorar este punto, se repite a continuación el experimento optimizando únicamente las relajaciones lineales con el solver de programación lineal de CPLEX.

En el Cuadro 8.2.2a, se muestran los resultados obtenidos. La fila "Gap (%)" corresponde al promedio del gap relativo entre el valor óptimo de la relajación lineal y el valor óptimo entero y "Tiempo (s)" es el promedio de los tiempos de ejecución medido en segundos. Vale la pena aclarar que el valor óptimo entero de todas las instancias de `Set2.1` es conocido por los resultados que serán reportados en la Sección 8.3. En la Figura 8.2.2b, se muestra un gráfico de barras que desagrega los resultados por instancia. Para cada instancia, numeradas del 1 al 15, se muestran tres barras con el gap relativo por formulación, en último lugar se muestran los valores promedios. La omisión de algunas de las formulaciones en la gráfica se debe a que las relajaciones lineales de LT-LT y LT-LTX tiene el mismo valor óptimo, lo mismo sucede con LTC-LT y LTC-LTX y con LTP-LT y LTP-LTX.

Considerando el valor óptimo de las relajaciones lineales, las formulaciones menos ajustadas son LT-LT y LT-LTX, le siguen LTC-LT y LTC-LTX y las mejores son LTP-LT y LTP-LTX, aunque el tiempo de ejecución va en aumento debido al mayor número de variables y restricciones. Estos resultados son consistentes con los anteriores, pues demuestran que las relajaciones lineales se vuelven ampliamente más ajustadas con la incorporación de mayor estructura en el digrafo que modela las rutas de los camiones y esto impacta directamente en la resolución de los programas enteros.

Formulación	LT-LT	LT-LTX	LTC-LT	LTC-LTX	LTP-LT	LTP-LTX
Gap (%)	95.4	95.4	51.9	51.9	11.1	11.1
Tiempo (s)	2.12	2.17	2.68	2.88	5.29	5.17

(a) Cuadro con el resumen de los resultados por formulación.



(b) Gráfico del gap relativo entre el valor óptimo de la relajación lineal y el valor óptimo entero por instancia y por formulación.

Cuadro 8.2.2.: Desempeño de las relajaciones lineales de las formulación sobre `Set2.1`.

Teniendo en cuenta los resultados obtenidos en este experimento, se decide continuar trabajando con las formulaciones LTC-LTX y LTP-LTX. Las formulaciones LT-LT y LT-LTX se abandonan debido a su mal desempeño, al igual que LTC-LT y LTP-LT debido a que a priori son ligeramente menos competitivas que LTC-LTX y LTP-LTX, respectivamente. El objetivo del estudio específico es fortalecer los algoritmos desarrollados hasta el momento y lograr resolver instancias de mayor tamaño.

8.3. Formulaciones alternativas

Las siguientes experiencias computacionales tienen por objetivo comparar el desempeño de las formulaciones LTC-LTX y LTP-LTX al reemplazar las restricciones originales por las alternativas presentadas en la Sección 7.1, es decir, reemplazar PREC_1 por PREC_2 (sólo aplica para LTC-LTX), DESC_1 por DESC_2 y SINC_1 por SINC_2 o por SINC_3. La metodología aplicada es la misma que en el experimento anterior.

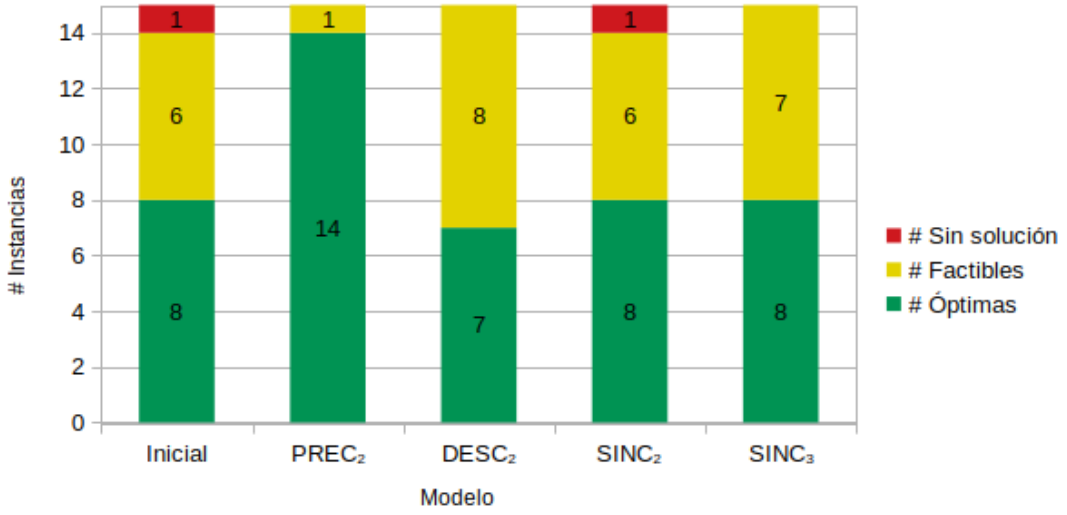
Formulaciones alternativas para LTC-LTX

Los resultados alcanzados por las formulaciones alternativas de LTC-LTX sobre el `Set2.1` se presentan en el Cuadro 8.3.1a. La columna "Inicial" repite los resultados del experimento anterior y en las columnas siguientes se reportan los resultados para las formulaciones alternativas obtenidas de reemplazar las restricciones originales por las que se corresponden con el nombre de la columna. Además, en la Figura 8.3.1b, se muestra un gráfico de barras con el número de instancias resueltas por formulación.

Los resultados obtenidos muestran una amplia mejoría al escribir las restricciones de precedencia en su versión alternativa PREC_2, donde se evidencia un marcado aumento en el número de instancias resueltas por optimalidad. Con respecto al resto de las restricciones alternativas, los resultados obtenidos son menos concluyentes. Al usar las restricciones DESC_2, una instancia menos es resuelta por optimalidad, y al usar SINC_2 se mantiene el mismo número de instancias resueltas, pero los tiempos de ejecución se incrementan. A pesar de que se mantiene

Formulación	LTC-LTX				
	Inicial	$PREC_2$	$DESC_2$	$SINC_2$	$SINC_3$
# Variables	10166	10166	10166	10166	10166
# Restricciones	6151	6151	6473	6151	5404
# Óptimas	8	14	7	8	8
# Factibles	6	1	8	6	7
# Sin solución	1	0	0	1	0
Tiempo (s)	4597	1274	4764	4674	4597
Tiempo opt. (s)	2319	851	1981	2463	2319
Nodos opt.	9154	7151	8468	105495	168612
Gap (%)	14	1,5	16	12	7
Gap fact. (%)	33	22	30	31	16

(a) Cuadro con el resumen de los resultados por formulación.



(b) Gráfico del número de instancias resueltas por formulación.

Cuadro 8.3.1.: Comparación de las formulaciones alternativas para LTC-LTX sobre `Set2.1`.

Formulación	LTC-LTX		
	$PREC_2$	$PREC_2$ + $SINC_2$	$PREC_2$ + $SINC_3$
# Variables	10166	10166	10166
# Restricciones	6151	6151	5404
# Óptimas	14	14	12
# Factibles	1	1	3
# Sin solución	0	0	0
Tiempo (s)	1274	1019	2201
Tiempo opt. (s)	851	577	951
Nodos opt.	7151	4611	54125
Gap (%)	1,5	3	0,6
Gap fact. (%)	22	39	3

Cuadro 8.3.2.: Comparación de las formulaciones alternativas para LTC-LTX con $PREC_2$ sobre `Set2.1`.

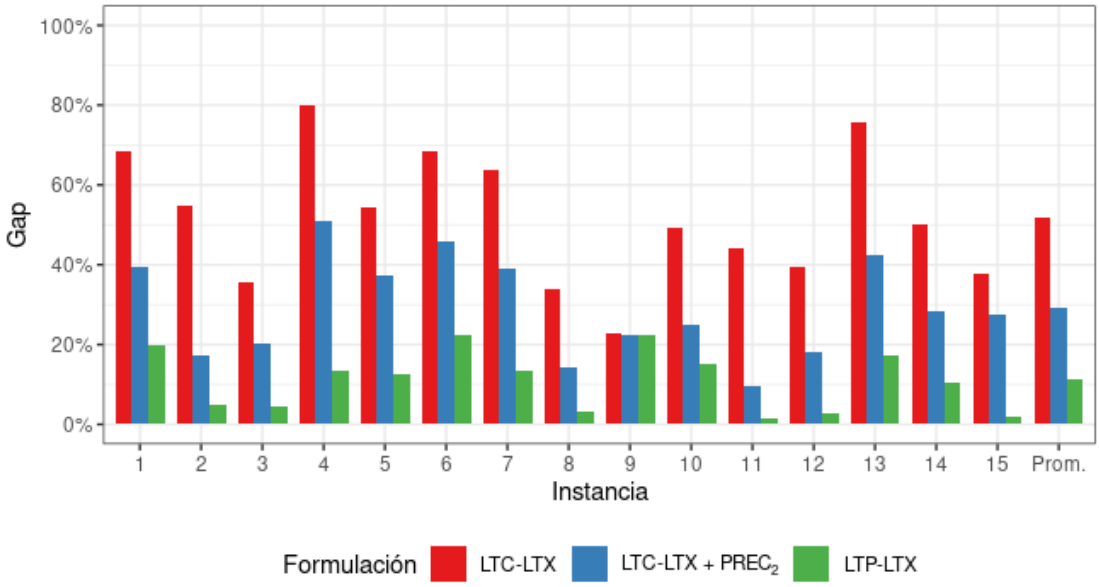
el mismo número de instancias resueltas por optimalidad con $SINC_3$, es posible encontrar una solución factible en la única instancia donde no se había encontrado y el gap se reduce a la mitad, pero el número de nodos aumenta en gran medida.

Es evidente que conviene reemplazar las restricciones de precedencia por las alternativas, pero no queda del todo claro cuáles restricciones de sincronización elegir. Por este motivo, se repite el experimento esta vez usando como formulación de base a LTC-LTX con $PREC_2$ y variando entre las diferentes alternativas para las restricciones de sincronización. Estos resultados se presentan en el Cuadro 8.3.2. Como puede observarse, la combinación $PREC_2$ + $SINC_2$ es la más competitiva, dado que mantiene el mismo número de instancias resueltas y mejora en gran medida los tiempos de ejecución y el número de nodos.

Los comportamientos observados se pueden explicar nuevamente por las diferencias en los valores óptimos de las relajaciones lineales. La relajación lineal de LTC-LTX se vuelve mucho más ajustada cuando se reemplazan las restricciones $PREC_1$ por $PREC_2$, y además se reducen ligeramente los tiempos de ejecución. Estos resultados se presentan en el Cuadro 8.3.3, donde se considera el gap relativo entre el valor óptimo de la relajación lineal y el de la solución óptima entera. A pesar de esta mejora, los valores óptimos de las relajaciones lineales aún no alcanzan a los de LTP-LTX. Vale la pena aclarar que las restricciones alternativas de sincronización se excluyen de este análisis porque no modificaron los valores

Formulación	LTC-LTX	LTC-LTX + $PREC_2$	LTP-LTX
Gap (%)	51	29	11
Tiempo (s)	2.88	2.57	5.17

(a) Cuadro con el resumen de los resultados por formulación.



(b) Gráfico del gap relativo entre el valor óptimo de la relajación lineal y el valor óptimo entero por instancia y por formulación.

Cuadro 8.3.3.: Desempeño de las relajaciones lineales de las formulaciones alternativas sobre `Set2.1`.

óptimos de las relajaciones lineales.

Formulaciones alternativas para LTP-LTX

Los resultados alcanzados por las formulaciones alternativas para LTP-LTX sobre el `Set2.1` se presentan en el Cuadro 8.3.4. Como se puede apreciar, las restricciones $DESC_2$ y $SINC_3$ son contraproducentes, dado que mantienen el mismo número de instancias resueltas pero con peores tiempos. Por el contrario, las restricciones $SINC_2$ permiten resolver una instancia adicional por optimalidad, logrando así resolver todas las instancias. En contrapartida, y considerando exclusivamente las 14 instancias en común resueltas por optimalidad, el promedio de los tiempos de

Formulación	LTP-LTX			
	Inicial	$DESC_2$	$SINC_2$	$SINC_3$
# Variables	16102	16102	16102	16102
# Restricciones	7828	8150	7828	7082
# Óptimas	14	14	15	14
# Factibles	1	1	0	1
# Sin solución	0	0	0	0
Tiempo (s)	894	969	917	2057
Tiempo opt. (s)	444	524	917	1690
Nodos opt.	1915	1443	2318	56299
Gap (%)	0,8	0,6	0	1,3
Gap fact. (%)	11	8	-	19

Cuadro 8.3.4.: Comparación de las formulaciones alternativas para LTP-LTX sobre `Set2.1`.

ejecución con $SINC_2$ es de 534 segundos, lo que significa un aumento de aproximadamente 20 %. Poniendo el foco en resolver por optimalidad el mayor número de instancias, la formulación más competitiva resulta entonces de reemplazar en LTP-LTX las restricciones $SINC_1$ por las $SINC_2$.

En base a lo expuesto en esta sección, se decide que los algoritmos de optimización planteados en el resto del capítulo estén basados en las siguientes formulaciones alternativas: (1) LTC-LTX con $PREC_2$ y $SINC_2$ y (2) LTP-LTX con $SINC_2$. A fin de facilitar la lectura, se referirá de ahora en adelante a estas formulaciones alternativas simplemente como LTC-LTX y LTP-LTX, respectivamente.

8.4. Heurística inicial

Una vez elegidas las formulaciones de base para usar en los algoritmos de optimización, a continuación se busca estudiar el impacto que tiene complementar el arranque con una heurística inicial. Contar con una solución factible desde el arranque de la optimización posibilita podar nodos por cota desde el comienzo en el árbol de branch-and-bound, sin necesidad de esperar a que el propio solver encuentre sus propias soluciones.

En el diseño de la heurística inicial, se aprovecharon los algoritmos de optimización secuenciales ya desarrollados para la primera parte de la tesis. Estos algoritmos

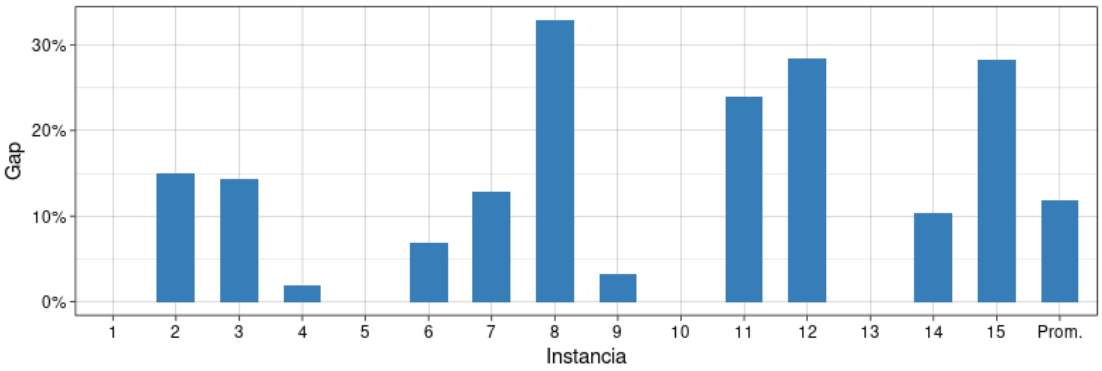


Figura 8.4.1.: Gráfica del gap relativo entre el valor de la solución heurística inicial y el valor óptimo entero para las instancias del `Set2.1`.

se aplican para cada instancia de la siguiente forma. En primer lugar, se generan 30 soluciones para el subproblema de la primera etapa, la mitad de ellas usando CPLEX y la otra mitad con la metaheurística híbrida LNS. Luego, para cada una de ellas, se resuelve el subproblema de la segunda etapa con la metaheurística GRASP×ILS y al finalizar se retorna la mejor solución entre todas. Se consideran los siguientes parámetros: CPLEX se llama con la funcionalidad *populate* para resolver por optimalidad la formulación E1 y retornar una solución óptima junto a 14 soluciones factibles que se encuentren a un gap del 3 % del valor óptimo, LNS se limita a 1000 iteraciones por instancia, GRASP×ILS se limita a 100 iteraciones por instancia y los demás parámetros conservan los valores elegidos en la primera parte de la tesis. Para las instancias aquí consideradas, el tiempo consumido por este procedimiento heurístico es relativamente bajo. En promedio, requiere 7 segundos por instancia para `Set2.1`, 18 segundos para `Set2.2` y `Set2.3` y 8 segundos para `Set2.4`. Sin embargo, la calidad de las soluciones heurísticas encontradas varía en gran medida. Por ejemplo, en la Figura 8.4.1, se puede ver una gráfica con la variación del gap relativo entre el valor de la solución heurística y el valor óptimo entero para cada una de las instancias del `Set2.1` (recordar que los valores óptimos enteros fueron encontrados para todas ellas en la sección anterior).

Se adaptaron los algoritmos de la sección anterior para que, durante el preprocesamiento de la instancia, las soluciones heurísticas sean incorporadas como iniciales. Esta tarea fue realizada por medio de la funcionalidad `mipstart` de CPLEX. Los resultados obtenidos para los algoritmos basados en LTC-LTX y LTP-LTX sobre el `Set2.1` se presentan en el Cuadro 8.4.1. Las columnas "PREC_2 + SINC_2" y

Algoritmo	LTC-LTX		LTP-LTX	
	$PREC_2$ + $SINC_2$	Heur. inicial	$SINC_2$	Heur. inicial
# Variables	10166	10166	16102	16102
# Restricciones	6151	6151	7828	7828
# Óptimas	14	14	15	15
# Factibles	1	1	0	0
Tiempo (s)	1019	931 (+7)	917	647 (+7)
Tiempo opt. (s)	577	484 (+7)	917	647 (+7)
Nodos opt.	4611	3518	2318	2286
Gap (%)	2,6	0,8	0	0
Gap fact. (%)	39	12	–	–

Cuadro 8.4.1.: Impacto de la heurística inicial sobre `Set2.1`.

"$SINC_2$"' repiten los resultados obtenidos en el experimento anterior y las columnas "Heur. inicial" presentan los nuevos resultados obtenidos al incorporar soluciones heurísticas iniciales. Las filas "Tiempo (s)" y "Tiempo opt. (s)" incluyen entre paréntesis el tiempo consumido por la heurística inicial.

Como se puede ver, contar con soluciones heurísticas iniciales resulta beneficioso. Si bien se mantiene el mismo número de instancias resueltas, los tiempos de ejecución se acortan y la cantidad de nodos explorados se reduce. Por estos motivos, se decide dejar incorporada la heurística inicial en los algoritmos de optimización durante el resto del capítulo.

8.5. Desigualdades válidas

Luego de complementar el arranque de los algoritmos de optimización con una heurística inicial para mejorar la cota primal, queda pendiente analizar la incorporación de desigualdades válidas para mejorar la cota dual proveniente de las relajaciones lineales. Las experiencias computacionales presentadas en esta sección evalúan de forma práctica las descriptas en el capítulo anterior.

Se propone la siguiente metodología de trabajo. En primer lugar, se estudia el impacto de las desigualdades válidas en la relajación lineal de las formulaciones, es decir, en el nodo raíz del árbol de branch-and-bound. En base a los resultados obtenidos, se eligen las familias más prometedoras y se estudia su incorporación

en los algoritmos de optimización para los programas enteros, es decir, se estudia el impacto sobre las relajaciones lineales de todos los nodos del árbol de branch-and-bound.

8.5.1. Relajación lineal en el nodo raíz

El primero de los análisis de esta sección tiene por objetivo medir el impacto de las desigualdades válidas en la relajación lineal de las formulaciones LTC-LTX y LTP-LTX. Para cada familia de desigualdades válidas, se aumenta la relajación lineal con todas las restricciones de esa familia y se analiza la variación en el valor óptimo fraccionario, el número de restricciones y el tiempo de ejecución.

En las pruebas se consideran todas las desigualdades válidas presentadas en la Sección 7.2. En algunas de estas familias aparecen involucrados subconjuntos de camiones o pedidos, lo que genera que el número de restricciones crezca exponencialmente con el tamaño de la instancia. Por este motivo, surge la necesidad de restringir su tamaño, de forma tal que sea posible generarlas exhaustivamente. Las consideraciones hechas se detallan a continuación.

- **RE$_3$**. Estas restricciones están expresadas para todo subconjunto de V. La forma tradicional de manejar este tipo de restricciones es limitarse a subfamilias **RE$_3$-V_k**, con $k \in \{1, \ldots, |V|\}$, donde los subconjuntos de camiones tienen cardinal a lo sumo k. En la práctica, esta limitación puede ser insuficiente (por ejemplo, RE$_3$-V_1 tiene un promedio de 214k restricciones en las instancias de `Set2.3`). A su vez, estas restricciones están cuantificadas universalmente para todo par de arcos e_1 de recolección y e_2 de entrega asociados a un mismo pedido p. Luego, se proponen las siguientes subfamilias de RE$_3$-V_k que restringen la elección de e_1 y e_2:

  - **RE$_3$-V_k-e_1**. Considera únicamente aquellas restricciones donde e_1 inicia exactamente en el primer instante de tiempo de cada ventana de tiempo de recolección de p.

  - **RE$_3$-V_k-e_2**. Considera únicamente aquellas restricciones donde e_2 inicia exactamente en el último instante de tiempo de cada ventana de tiempo de entrega de p.

- **RE$_3$-V_k-$e_\cap$** y **RE$_3$-V_k-$e_\cup$**. Son respectivamente la intersección y la unión entre RE$_3$-V_k-e_1 y RE$_3$-V_k-e_2.

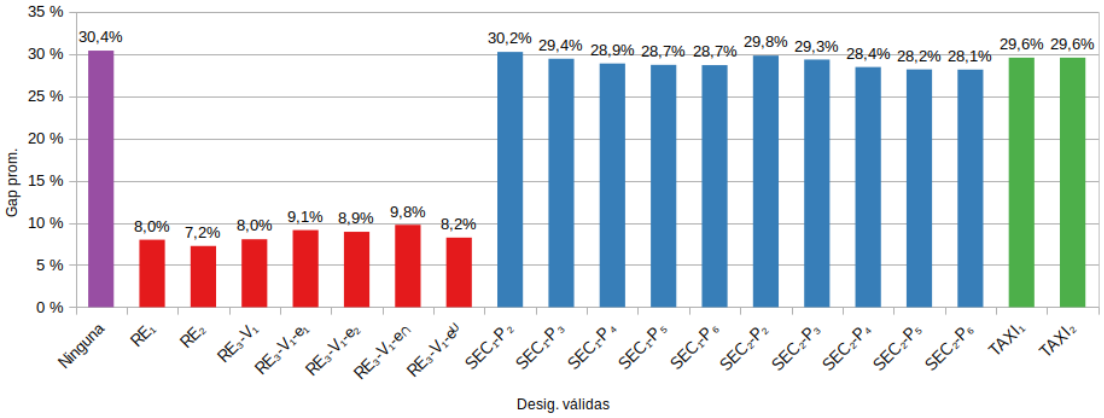
- **SEC$_1^{\mathrm{R}}$** y **SEC$_1^{\mathrm{E}}$** (y análogamente para **SEC$_2^{\mathrm{R}}$** y **SEC$_2^{\mathrm{E}}$**). En estas restricciones intervienen subconjuntos de P. Para limitar su tamaño se propone considerar las subfamilias **SEC$_1^{\mathrm{R}}$-P_k** y **SEC$_1^{\mathrm{E}}$-P_k**, con $k \in \{2, \ldots, |P|\}$, constituidas por todas las restricciones donde el subconjunto de pedidos tiene cardinal a lo sumo k. También se considera la subfamilia **SEC$_1$-P_k** formada por la unión entre SEC$_1^{\mathrm{R}}$-P_k y SEC$_1^{\mathrm{E}}$-P_k. Para cada subconjunto de pedidos P', los valores de los parámetros *sdur* se computan evaluando exhaustivamente todas las permutaciones de P' (para los valores de k considerados en las pruebas, esta evaluación es sumamente rápida).

En el Cuadro 8.5.1, se muestran los resultados obtenidos al resolver la relajación lineal aumentada de LTC-LTX. En este caso, se consideran las instancias del `Set2.2`, las cuales poseen mayor número de pedidos. Cada fila del Cuadro 8.5.1a resume los resultados promedios para la familia de desigualdades válidas indicada en la columna "Desig. válidas". La columna "RL" muestra el promedio del valor óptimo de la relajación lineal aumentada, "Gap (%)" es el promedio del porcentaje de gap relativo entre el valor óptimo de la relajación lineal aumentada y el valor óptimo entero (el cual se conoce por los resultados que se verán en la siguiente subsección), "# Desig." es el promedio del número desigualdades válidas de la familia y "Tiempo (s)" es el promedio del tiempo de ejecución en segundos. Además, en la Figura 8.5.1b se muestra un gráfico de barras del promedio de gap relativo por familia de desigualdades válidas.

Es evidente que todas las variantes de las restricciones de viaje recolección-entrega son altamente efectivas en reforzar las relajaciones lineales, obteniendo un promedio de gap que se ubica entre el 7,2 % y el 9,8 %, en comparación con el 30,4 % que posee la formulación de base. Entre ellas, las RE$_2$ obtienen el promedio de gap más bajo, seguidas por las RE$_1$, las RE$_3$-V_1 y las RE$_3$-V_1-$e_\cup$, aunque las últimas dos hacen que las formulaciones sean mucho más pesadas. Por su parte, las RE$_3$-V_1-$e_\cap$ presentan promedios de gaps un poco más altos que las demás restricciones RE, pero se destacan por su bajo número de desigualdades y tiempos de ejecución. Además, es llamativo que el promedio de tiempo de ejecución para las RE$_3$-V_1-$e_\cup$ sea mayor que para las RE$_3$-V_1, dado que son una subfamilia. Respecto a las

Desig. Válidas	RL	Gap (%)	# Desig.	Tiempo (s)
Ninguna	59,9	30,4	0	4,0
RE_1	78,6	8,0	6	7,4
RE_2	79,2	7,2	686	9,5
RE_3-V_1	78,4	8,0	55275	38,6
RE_3-V_1-e_1	77,5	9,1	4339	18,9
RE_3-V_1-e_2	77,7	8,9	4541	23,3
RE_3-V_1-$e_\cap$	77,0	9,8	349	7,1
RE_3-V_1-$e_\cup$	78,3	8,2	8531	54,3
SEC_1-P_2	60,0	30,2	30	3,9
SEC_1-P_3	60,7	29,4	107	4,1
SEC_1-P_4	61,2	28,9	223	4,5
SEC_1-P_5	61,4	28,7	335	4,6
SEC_1-P_6	61,4	28,7	406	4,9
SEC_2-P_2	60,4	29,8	6934	6,5
SEC_2-P_3	60,8	29,3	23233	11,8
SEC_2-P_4	61,6	28,4	45670	14,1
SEC_2-P_5	61,8	28,2	65948	19,0
SEC_2-P_6	61,9	28,1	78140	22,6
$TAXI_1$	60,6	29,6	3	3,9
$TAXI_2$	60,6	29,6	236	4,4

(a) Cuadro con los resultados promedios por familia de desigualdades válidas.



(b) Gráfico del gap relativo entre el valor óptimo de la relajación lineal y el valor óptimo entero por familia de desigualdades válidas.

Cuadro 8.5.1.: Desempeño de las relajaciones lineales de LTC-LTX aumentadas con las desigualdades válidas sobre `Set2.2`.

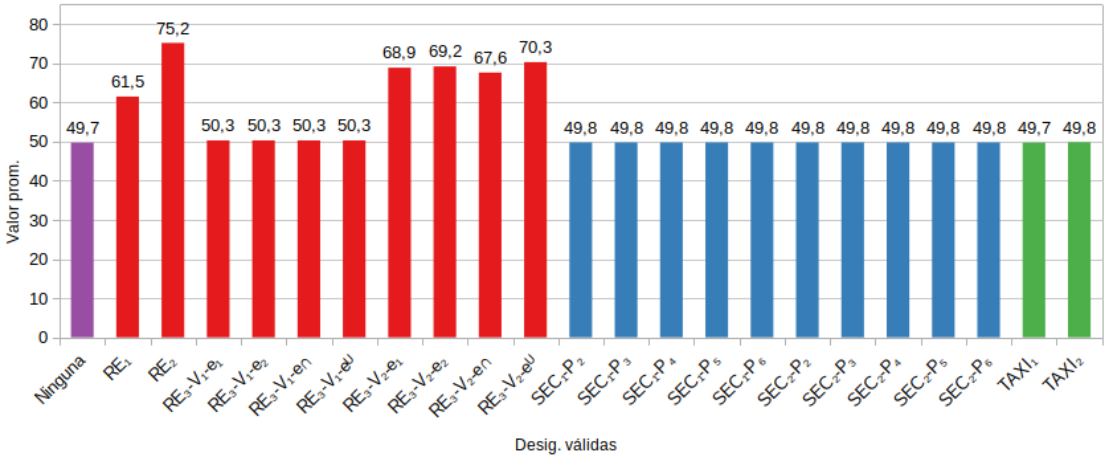
restricciones de secuenciación de pedidos, el ajuste de las relajaciones lineales es más moderado, visto que el promedio de gap se ubica entre el 28,1 % y el 30,2 %. Tanto para las SEC$_1$-P_k, como para las SEC$_2$-P_k, el promedio de gap es menor cuando mayor es k, sin embargo la mejora es cada vez menos notoria a partir $k = 4$. Los promedios de gap para las SEC$_2$-P_k son ligeramente mejores que para las SEC$_1$-P_k, aunque se evidencia un gran aumento en el número de desigualdades y en los tiempos de ejecución. También hay que mencionar que las SEC$_1$-P_k logran mejorar el promedio de gap que tienen individualmente las SEC$_1^{\text{R}}$-P_k y las SEC$_1^{\text{E}}$-P_k, y lo mismo ocurre con las SEC$_2$-P_k, motivo por el cual se excluyen del cuadro estas familias individuales. Por último, con las restricciones de viaje en taxi, ambas disminuyen ligeramente el promedio de gap a 29,6 %, con tiempos de ejecución casi idénticos a la formulación de base, pero las TAXI$_2$ tienen muchas más desigualdades que las TAXI$_1$.

Todas estas desigualdades válidas también pueden ser incorporadas en LTC-LTX cuando hay múltiples camiones, donde además tiene sentido considerar las restricciones RE$_3$-V_k con $k \geq 2$. Por este motivo, se repite el experimento con el `Set2.3` a fin de analizar su impacto en instancias con más de un camión. Los resultados obtenidos se muestran en el Cuadro 8.5.2, en los cuales no se reporta el gap debido a que no se conocen soluciones óptimas enteras para todas las instancias. En el gráfico de barras de la Figura 8.5.2b, se muestra el promedio de los valores óptimos de las relajaciones lineales aumentadas para cada familia de desigualdades válidas. En este caso, no se presentan resultados para las RE$_3$-V_1 y las RE$_3$-V_2 debido a que involucran demasiadas desigualdades y por falta de memoria no es posible resolver algunas de las instancias.

Se puede observar que las variantes de las restricciones de viaje recolección-entrega nuevamente son las que mayor impacto generan a nivel relajación lineal. Otra vez, las RE$_2$ alcanzan en promedio los óptimos fraccionarios con mayor valor objetivo, seguidas por las RE$_3$-V_2-$e_\cup$, aunque la diferencia entre ellas es significativa. Es interesante notar que las subfamilias de RE$_3$-V_k casi no tienen impacto cuando $k = 1$, pero se vuelven muy efectivas cuando $k = 2$, incluso llegan a superar los valores promedios alcanzados por las RE$_1$. Otro punto llamativo es el gran aumento en los tiempos de ejecución que tienen las RE$_3$-V_k-e_1 y las RE$_3$-V_k-e_2 cuando se pasa de $k = 1$ a $k = 2$, en contraste con las RE$_3$-V_k-$e_\cap$ y las RE$_3$-V_k-$e_\cup$, donde el incremento es menos abrupto. Por otro lado, ninguna de las restricciones

Desig. Válidas	RL	# Desig.	Tiempo (s)
Ninguna	49,7	0	7,4
RE_1	61,5	12	14,6
RE_2	75,2	1403	24,9
RE_3-V_1-e_1	50,3	9696	38,5
RE_3-V_1-e_2	50,3	9366	39,8
RE_3-V_1-$e_\cap$	50,3	789	10,3
RE_3-V_1-$e_\cup$	50,3	18273	142,3
RE_3-V_2-e_1	68,9	14544	135,2
RE_3-V_2-e_2	69,2	14049	145,7
RE_3-V_2-$e_\cap$	67,6	1184	17,4
RE_3-V_2-$e_\cup$	70,3	27409	147,7
SEC_1-P_2	49,8	85	7,5
SEC_1-P_3	49,8	335	8,3
SEC_1-P_4	49,8	770	8,7
SEC_1-P_5	49,8	1272	9,3
SEC_1-P_6	49,8	1670	10,5
SEC_2-P_2	49,8	18507	18,0
SEC_2-P_3	49,8	68120	33,4
SEC_2-P_4	49,8	148341	46,3
SEC_2-P_5	49,8	235924	79,6
SEC_2-P_6	49,8	301863	115,6
$TAXI_1$	49,7	4	7,5
$TAXI_2$	49,8	317	8,8

(a) Cuadro con los resultados promedios por familia de desigualdades válidas.



(b) Gráfico del promedio del valor óptimo de la relajación lineal por familia de desigualdades válidas.

Cuadro 8.5.2.: Desempeño de las relajaciones lineales de LTC-LTX aumentadas con las desigualdades válidas sobre `Set2.3`.

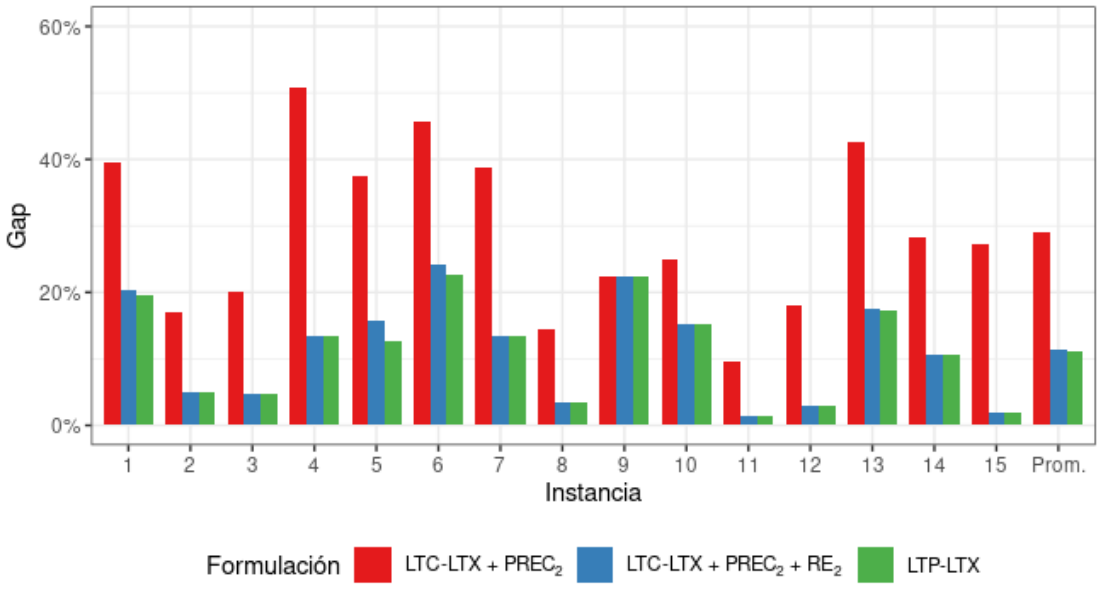


Figura 8.5.1.: Gráfico del gap relativo entre el valor óptimo de la relajación lineal y el valor óptimo entero en las instancias de `Set2.1` para algunas de las formulaciones.

de secuenciación de pedidos, ni de viaje en taxi, tiene un impacto significativo en los valores óptimos de las relajaciones lineales. Estos resultados sugieren que estas desigualdades válidas no son útiles para elevar la cota dual en instancias con múltiples camiones, pero podrían ser útiles para cortar soluciones fraccionarias. Algunas ideas preliminares para afrontar estos inconvenientes fueron mencionados en la Sección 7.2.4, aunque todavía se requiere un estudio más profundo de las mismas.

Es importante destacar que las restricciones RE_2 son de hecho tan efectivas que muchas veces igualan los valores óptimos alcanzados por las relajaciones lineales de la formulación LTP-LTX, como se puede ver en el gráfico de barras de la Figura 8.5.1 para las instancias del `Set2.1`. Los valores reportados para LTC-LTX con PREC_2 y para LTP-LTX son los mismos que aparecen en el Cuadro 8.3.3.

Por último, en el Cuadro 8.5.3 se muestran los resultados obtenidos al resolver las relajaciones lineales aumentadas de LTP-LTX para las instancias del `Set2.2`. Vale la pena recordar que las restricciones de viaje recolección-entrega no se deben considerar, dado que las soluciones fraccionarias que intentan cortar no pueden existir por la estructura del digrafo G_{LTP}. En este caso, los comportamientos que

se observan son muy similares a los mencionados en el análisis de LTC-LTX, salvo dos grandes diferencias. Por un lado, los valores óptimos de las relajaciones lineales de LTP-LTX son de partida mucho más altos que los de LTC-LTX, motivo por el cual los gaps son mucho más ajustados. Por el otro lado, se observan mayores tiempos de ejecución. Este experimento no se repite para el `Set2.3` debido a que en esta formulación las restricciones de secuenciación de pedidos y de viaje en taxi no se pueden extender a instancias con múltiples camiones.

Los resultados alcanzados por las otras desigualdades válidas de la Sección 7.2.4 fueron excluidos de los cuadros anteriores debido a su bajo impacto en comparación a las demás. A partir de este análisis sobre el nodo raíz, se decide que las familias de desigualdades válidas RE_1, RE_2, RE_3-V_2-$e_\cup$ y RE_3-V_2-$e_\cap$ son las más prometedoras para continuar estudiando, seguidas por SEC_1-P_4 y SEC_2-P_4 y luego TAXI_1 y TAXI_2. En lo que sigue, se analiza su impacto general sobre los nodos de todo el árbol de branch-and-bound.

8.5.2. Relajaciones lineales en los nodos del árbol

El segundo de los análisis de esta sección tiene por objetivo medir el impacto de las desigualdades válidas sobre las relajaciones lineales de todos los nodos del árbol de branch-and-bound. Existen diversas estrategias para incorporarlas:

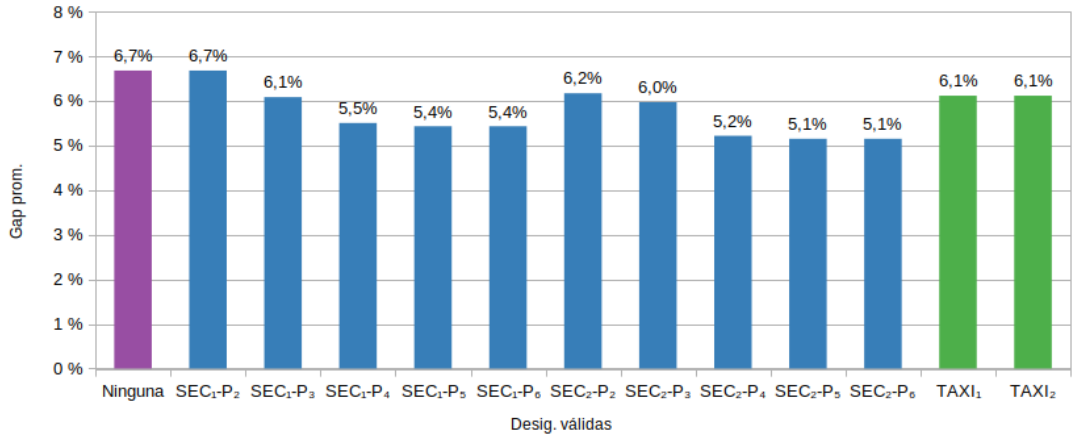
Restricciones en la formulación (RF). Agregarlas inicialmente como restricciones de la formulación, es decir, las relajaciones lineales de todos los nodos del árbol ya contienen a todas las desigualdades válidas de la familia considerada.

Pool de cortes (PC). Mantenerlas en un *pool de cortes* y dejar que el solver las incorpore bajo demanda según sus propios criterios en las relajaciones lineales de los nodos del árbol.

Callback de cortes (CB). Proveerle al solver una rutina de separación propia mediante un *callback de corte*. Por ejemplo, la rutina más sencilla consiste en tener todas las desigualdades válidas generadas de antemano y evaluarlas secuencialmente buscando aquellas que estén violadas por más de un determinado valor umbral. Por supuesto, también es posible definir rutinas mucho más sofisticadas, las cuales pueden ser tanto heurísticas como exactas.

Desig. Válidas	RL	Gap (%)	# Desig.	Tiempo (s)
Ninguna	79,7	6,7	0	8,1
SEC_1-P_2	79,7	6,7	30	8,7
SEC_1-P_3	80,1	6,1	107	9,4
SEC_1-P_4	80,7	5,5	223	8,2
SEC_1-P_5	80,7	5,4	335	8,3
SEC_1-P_6	80,7	5,4	406	8,5
SEC_2-P_2	80,1	6,2	6934	13,7
SEC_2-P_3	80,3	6,0	23233	30,7
SEC_2-P_4	80,9	5,2	45670	33,4
SEC_2-P_5	81,0	5,1	65948	36,9
SEC_2-P_6	81,0	5,1	65948	36,9
$TAXI_1$	80,0	6,1	3	8,1
$TAXI_2$	80,0	6,1	236	9,0

(a) Cuadro con los resultados promedios por familia de desigualdades válidas.



(b) Gráfico del gap relativo entre el valor óptimo de la relajación lineal y el valor óptimo entero por familia de desigualdades válidas.

Cuadro 8.5.3.: Desempeño de las relajaciones lineales de LTP-LTX aumentadas con las desigualdades válidas sobre `Set2.2`.

La elección de la estrategia más conveniente para manejar cada familia de desigualdades válidas depende de muchos factores. Por ejemplo, RF podría provocar que las relajaciones lineales se pongan demasiado pesadas al agregar todas las desigualdades válidas de una familia muy grande, en cuyo caso PC o CB podrían ser más convenientes ya que las manejarían bajo demanda. En las experiencias computacionales llevadas a cabo, se utilizará la estrategia RF cuando la familia está conformada por pocas desigualdades válidas, PC cuando la familia sea demasiado grande y ambas para algunos casos intermedios. Por su parte, CB no será considerada debido a que en este trabajo no fueron desarrollados algoritmos de separación específicos que sean más eficientes que una simple evaluación secuencial de todas las desigualdades válidas, que hasta donde entendemos sería similar a la rutina que aplica internamente PC. Durante las siguientes experiencias computacionales, todos los parámetros asociados con los cortes específicos del solver son mantenidos con sus valores por defecto.

Restricciones de viaje recolección-entrega

En primer lugar, se analiza la incorporación de las familias de desigualdades válidas RE_1, RE_2, RE_3-V_2-$e_\cap$ y RE_3-V_2-$e_\cup$ sobre el mejor algoritmo de optimización obtenido hasta ahora para la formulación LTC-LTX.

En el Cuadro 8.5.4 se muestran los resultados obtenidos para las instancias del `Set2.2`, `Set2.3` y `Set2.4`. El formato sigue en líneas generales al de los ya presentados. La segunda columna de "LTC-LTX" repite los resultados anteriores y las subsiguientes muestran los nuevos resultados. La estrategia usada para incorporar cada familia de desigualdades (RF o PC) se escribe entre paréntesis junto al nombre de cada algoritmo. Se recomienda consultar el Cuadro 8.1.1 para obtener una descripción completa de la configuración usada en cada algoritmo.

A diferencia de los cuadros anteriores, se agregan dos nuevas filas: "# Desig." es el promedio del número de desigualdades válidas que componen la familia considerada y "# Cortes" es el promedio del número de cortes de usuario agregados por el solver durante la optimización cuando la estrategia es PC. En general, todos los valores se calculan sumando o promediando sobre todas las instancias (en total 45), salvo en las filas correspondientes a "# Óptimas" y "Tiempo (s)", en las cuales los valores

se desagregan por conjunto de instancias en las primeras tres filas y se computan sobre todas las instancias en la última fila.

Se puede advertir un claro mejoramiento en la performance, independientemente de la familia de desigualdades válidas que se incorpore. En todos los casos, aumenta el número de instancias resueltas por optimalidad, a excepción de las RE_1 para el `Set2.4`, y se reducen los promedios de tiempo en que se prueba optimalidad y de gap relativo. Entre ellas, los mejores resultados se dan con las RE_2, seguidas por las RE_3 y en el último lugar quedan las RE_1; este comportamiento es consistente con lo observado anteriormente para el nodo raíz. Particularmente para las RE_2, las dos estrategias consideradas para su incorporación tienen puntos a favor. Por un lado, con PC se reduce ampliamente el promedio de tiempo en que se prueba optimalidad en el `Set2.2`. Por el otro lado, con RF es posible resolver una instancia más por optimalidad en el `Set2.3` y el promedio de tiempo en que se prueba optimalidad entre todas las instancias es ligeramente menor, aproximadamente en un 3 %. Por estos motivos, se decide dejar activadas las restricciones RE_2 con la estrategia RF para los próximos experimentos computacionales. De todas formas, la otra estrategia también es una opción interesante para continuar estudiando a futuro.

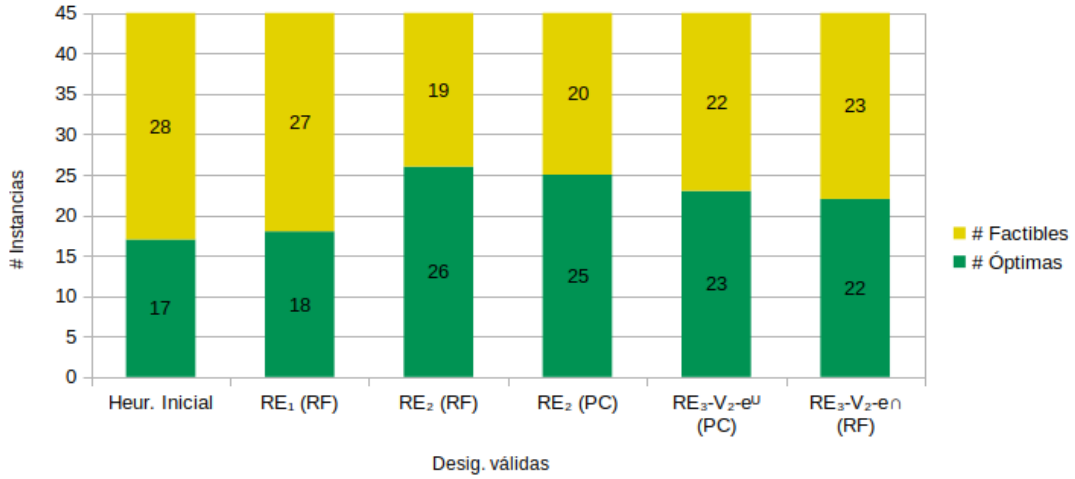
Restricciones de secuenciación de pedidos

A continuación, se analiza el desempeño de los algoritmos de optimización cuando se integran las desigualdades válidas SEC_1-P_4 y SEC_2-P_4. En los Cuadros 8.5.5 y 8.5.6 se resumen los resultados obtenidos por los algoritmos basados en las formulaciones LTC-LTX y LTP-LTX, respectivamente, sobre las instancias del `Set2.2` y `Set2.4`. El `Set2.3` no es considerado por dos motivos: las experiencias computacionales sobre el nodo raíz mostraron que no son efectivas para elevar el valor óptimo de la relajación lineal de LTC-LTX y pueden expresarse únicamente en instancias de LTP-LTX con un único camión.

En el algoritmo basado en LTC-LTX, se aprecia que algunos indicadores mejoran y otros empeoran. En todos los casos, una instancia que anteriormente podía ser resuelta por optimalidad, ahora pasa a ser reportada como factible. Entre lo positivo, se destaca la disminución en el promedio de tiempo de ejecución total

Algoritmo		LTC-LTX					
		Heur. inicial	RE_1 (RF)	RE_2 (RF)	RE_2 (PC)	RE_3-V_2-$e_\cup$ (PC)	RE_3-V_2-$e_\cap$ (RF)
# Variables		20647	20647	20647	20647	20647	20647
# Restricciones		10100	10110	11013	10100	10100	10687
# Óptimas	Set2.2	10	12	13	13	13	12
	Set2.3	0	1	6	5	3	3
	Set2.4	7	5	7	7	7	7
		17	18	26	25	23	22
# Factibles		28	27	19	20	22	23
Tiempo (s)	Set2.2	3794	3174	2747	1972	2353	2798
	Set2.3	7200	7036	5398	6290	6361	6271
	Set2.4	5429	5780	5231	5555	5126	4826
		5474	5330	4459	4606	4614	4632
Tiempo opt. (s)		2632	2525	2456	2531	2140	1946
Nodos opt.		8415	42325	25916	6438	5065	6206
Gap (%)		18	13	6	6	9	10
Gap fact. (%)		29	21	15	13	17	19
# Desig.		–	10	912	912	13837	587
# Cortes		–	–	–	115	379	–

(a) Cuadro con los resultados promedios por familia de desigualdades válidas.



(b) Gráfica con el número de instancias resueltas por familia de desigualdades válidas.

Cuadro 8.5.4.: Desempeño del algoritmo de optimización basado en LTC-LTX al incorporar las desigualdades válidas de viaje recolección-entrega, sobre las instancias de `Set2.2`, `Set2.3` y `Set2.4`.

Algoritmo		LTC-LTX			
		RE_2 (RF)	SEC_1-P_4 (RF)	SEC_1-P_4 (PC)	SEC_2-P_4 (PC)
# Variables		18906	18906	18906	18906
# Restricciones		10789	10918	10789	10789
# Óptimas	`Set2.2`	13	13	13	12
	`Set2.4`	7	6	6	7
		20	19	19	19
# Factibles		10	11	11	11
Tiempo (s)	`Set2.2`	2747	2661	2791	2651
	`Set2.4`	5231	5103	5610	5447
		3989	3882	4201	4049
Tiempo opt. (s)		2384	1961	2464	2225
Nodos opt.		28398	13447	19197	13601
Gap (%)		7	8	8	8
Gap fact. (%)		22	21	22	22
# Desig.		–	129	129	26427
# Cortes		–	–	0,5	3

Cuadro 8.5.5.: Desempeño del algoritmo de optimización basado en LTC-LTX al incorporar las desigualdades válidas de secuenciación de pedidos, sobre las instancias de `Set2.2` y `Set2.4`.

que se registra cuando se activan las SEC_1-P_4 con la estrategia RF. También se observa que las desigualdades válidas no impactan en el promedio de gap relativo y que el promedio de número de cortes agregados es muy bajo para la estrategia PC. Debido a que estos resultados no son del todo concluyentes, se decide omitir la incorporación de estas desigualdades válidas en las próximas experiencias computacionales que involucren a este algoritmo.

En el algoritmo basado en LTP-LTX, se evidencia una mejora en su desempeño cuando se incorporan las SEC_1-P_4 con la estrategia PC. Es posible resolver una instancia adicional por optimalidad, mejora el promedio de tiempo total de ejecución y se preserva el promedio de tiempo de ejecución en las instancias resueltas por optimalidad. Con las SEC_1-P_4 (RF) y las SEC_2-P_4, el número de instancias resueltas por optimalidad disminuye en una unidad respecto del original. En este caso, se puede ver que las desigualdades válidas sí generan una disminución en el promedio de gap relativo, pero que el promedio de número de cortes agregados sigue siendo bajo para la estrategia PC. En base a estos resultados, se decide que este algoritmo mantenga activadas las SEC_1-P_4 con la estrategia PC para los

Algoritmo		LTP-LTX			
		Heur. inicial	SEC_1-P_4 (RF)	SEC_1-P_4 (PC)	SEC_2-P_4 (PC)
# Variables		32664	32664	32664	32664
# Restricciones		13477	13607	13477	13477
# Óptimas	`Set2.2`	15	14	15	14
	`Set2.4`	8	8	9	8
		23	22	24	22
# Factibles		7	8	6	8
Tiempo (s)	`Set2.2`	1375	1619	1143	1136
	`Set2.4`	3929	4306	3761	3961
		2652	2963	2452	2548
Tiempo opt. (s)		1268	1422	1265	857
Nodos opt.		4834	42406	4008	3537
Gap (%)		4	3	2	3
Gap fact. (%)		15	11	12	12
# Desig.		–	129	129	26427
# Cortes		–	–	0,5	2

Cuadro 8.5.6.: Desempeño del algoritmo de optimización basado en LTP-LTX al incorporar las desigualdades válidas de secuenciación de pedidos, sobre las instancias de `Set2.2` y `Set2.4`.

Algoritmo		LTC-LTX			
		RE_2 (RF)	$TAXI_1$ (RF)	$TAXI_2$ (RF)	$TAXI_2$ (PC)
# Variables		18906	18906	18906	18906
# Restricciones		10789	10791	11034	10789
# Óptimas	`Set2.2`	13	13	13	13
	`Set2.4`	7	7	7	7
		20	20	20	20
# Factibles		10	10	10	10
Tiempo (s)	`Set2.2`	2747	2388	2459	2530
	`Set2.4`	5231	5235	5664	5381
		3989	3811	4061	3955
Tiempo opt. (s)		2384	2117	2492	2333
Nodos opt.		28398	18493	19498	18690
Gap (%)		7	8	8	7
Gap fact. (%)		22	24	23	21
# Desig.		–	3	245	245
# Cortes		–	–	–	32

Cuadro 8.5.7.: Desempeño del algoritmo de optimización basado en LTC-LTX al incorporar las desigualdades válidas de viaje en taxi, sobre las instancias de `Set2.2` y `Set2.4`.

próximos experimentos computacionales.

Restricciones de viaje en taxi

Por último, se evalúa el desempeño de los algoritmos de optimización con las desigualdades válidas $TAXI_1$ y $TAXI_2$. En los Cuadros 8.5.7 y 8.5.8 se reportan los resultados obtenidos por los algoritmos basados en las formulaciones LTC-LTX y LTP-LTX, respectivamente, sobre las instancias del `Set2.2` y `Set2.4` (el `Set2.3` no se considera por los motivos ya mencionados). En el caso de LTP-LTX, hay que tener en cuenta que la fila "# Cortes" contabiliza también a los SEC_1-P_4, debido a que el solver no permite separar el número de cortes agregados por familia.

En el algoritmo basado en LTC-LTX, se observan los mismos números de instancias resueltas, pero variaciones en los tiempos de ejecución. En particular, todas las desigualdades válidas logran reducir el promedio del tiempo de ejecución para el `Set2.2`, siendo las $TAXI_1$ las que alcanzan los mejores resultados. Sin embargo,

Algoritmo		LTP-LTX			
		SEC_1-P_4 (PC)	$TAXI_1$ (RF)	$TAXI_2$ (RF)	$TAXI_2$ (PC)
# Variables		32664	32664	32664	32664
# Restricciones		13477	13480	13723	13477
# Óptimas	`Set2.2`	15	15	15	15
	`Set2.4`	9	9	9	10
		24	24	24	25
# Factibles		6	6	6	5
Tiempo (s)	`Set2.2`	1143	1273	1343	976
	`Set2.4`	3761	3714	3954	3352
		2452	2493	2649	2164
Tiempo opt. (s)		1265	1317	1511	1157
Nodos opt.		4008	3637	3968	3564
Gap (%)		2	4	4	3
Gap fact. (%)		12	18	19	16
# Desig.		–	3	245	245
# Cortes		0,5	0,6	0,5	23

Cuadro 8.5.8.: Desempeño del algoritmo de optimización basado en LTP-LTX al incorporar las desigualdades válidas de viaje en taxi, sobre las instancias de `Set2.2` y `Set2.4`.

para el `Set2.4`, el promedio del tiempo de ejecución se mantiene igual con las TAXI_1 y se incrementa con las TAXI_2. En base a esto, resulta conveniente la incorporación de las TAXI_1 con la estrategia RF.

En el algoritmo basado en LTP-LTX, se destacan los resultados obtenidos por las TAXI_2 con la estrategia PC. En particular, es posible resolver una instancia adicional por optimalidad y el promedio de los tiempos de ejecución se reduce notoriamente. Por el contrario, las TAXI_1 y las TAXI_2 (RF) no parecen ser efectivas, ya que si bien mantienen el mismo número de instancias resueltas, los tiempos de ejecución aumentan. A su vez, llama la atención el aumento que se observa en el promedio de gap relativo al incluir estas desigualdades.

8.6. Comparaciones finales y limitaciones

En esta sección se muestran los resultados finales conseguidos por ambos algoritmos al incluir todos los desarrollos que demostraron ser efectivos anteriormente. En el caso del algoritmo basado en LTC-LTX, se intercambian de la formulación las restricciones PREC_1 y SINC_1 por las PREC_2 y SINC_2, respectivamente, se provee una solución heurística inicial y se incorporan a la formulación las desigualdades válidas RE_2 y TAXI_1. En el caso del algoritmo basado en LTP-LTX, se intercambian de la formulación las restricciones SINC_1 por las SINC_2, se provee una solución heurística inicial y se agregan a un pool de cortes las desigualdades válidas de SEC_1-P_4 y TAXI_2.

En el Cuadro 8.6.1 se ven los resultados conseguidos para cada una de las instancias del `Set2.2`, `Set2.3` y `Set2.4`. En cada fila, se resaltan en negrita el mejor tiempo y gap relativo alcanzados entre los dos algoritmos. A su vez, en el Cuadro 8.6.2 se exhiben los valores promedios entre todas las instancias.

Es evidente que el algoritmo basado en LTP-LTX logra resolver un mayor número de instancias por optimalidad, 37 contra 26. Incluso, todas las instancias que LTC-LTX resuelve por optimalidad, también son resueltas por LTP-LTX y más rápidamente (salvo 2 de ellas), con un promedio de tiempo de ejecución de 734s contra 2250s. Además, en los casos en donde ninguno de los dos logra probar optimalidad, LTP-LTX reporta mejor gap relativo (salvo en 1 instancia), con

$\|L\|$	$\|R\|$	$\|V\|$	$\|C\|$	LTC-LTX				LTP-LTX			
				Tiempo (s)	Nodos opt.	Cota inf.	Gap (%)	Tiempo (s)	Nodos opt.	Cota inf.	Gap (%)
3	7	1	2	2119	2147	57,0	0,0	**1152**	2229	57,0	0,0
				7200	3844	71,0	11,3	**6832**	12053	75,0	**0,0**
				409	7118	68,0	0,0	**100**	2286	68,0	0,0
				804	1864	73,0	0,0	**72**	2113	73,0	0,0
				2581	3850	70,0	0,0	**666**	2428	70,0	0,0
3	8	1	2	4553	3037	92,0	0,0	**1336**	1328	92,0	0,0
				1441	3332	103,0	0,0	**1053**	13373	103,0	0,0
				3428	156970	77,0	0,0	**326**	10511	77,0	0,0
				1352	146256	91,0	0,0	**99**	6233	91,0	0,0
				219	4193	82,0	0,0	**24**	2212	82,0	0,0
3	9	1	2	374	2290	69,0	0,0	**151**	1145	69,0	0,0
				1416	3368	102,0	0,0	**827**	3125	102,0	0,0
				7200	3542	101,0	9,8	**315**	2016	102,0	**0,0**
				2317	11480	87,0	0,0	**1411**	2425	87,0	0,0
				402	12731	131,0	0,0	**272**	13150	131,0	0,0
3	8	2	4	5538	3243	55,0	0,0	**574**	1367	55,0	0,0
				1566	228	75,0	0,0	**995**	1532	75,0	0,0
				7200	347905	100,0	1,0	**5021**	306871	101,0	**0,0**
				676	133	77,0	0,0	**594**	10271	77,0	0,0
				7200	58008	66,0	1,5	**2218**	169018	67,0	**0,0**
3	9	2	4	7200	2436	52,0	14,8	**3042**	10002	61,0	**0,0**
				7200	302	103,8	15,0	7200	1595	104,0	**13,3**
				2304	79979	70,0	0,0	**192**	5089	70,0	0,0
				2696	19177	69,0	0,0	**643**	2053	69,0	0,0
				7200	17494	86,0	3,4	**4632**	29264	87,0	**0,0**
3	10	2	4	3389	3095	76,0	0,0	**443**	8720	76,0	0,0
				7200	2319	69,0	11,5	**3631**	15984	71,0	**0,0**
				7200	2585	99,0	2,9	7200	654913	99,0	**1,0**
				7200	1202	72,1	8,7	**3466**	1412	78,0	**0,0**
				7200	29797	89,0	**1,1**	7200	4046	88,2	2,0
6	4	1	2	7200	515	50,2	28,3	7200	1434	68,0	**17,1**
				3427	1717	48,0	0,0	**1718**	1051	48,0	0,0
				39	0	42,0	0,0	218	44	42,0	0,0
				3010	1656	46,0	0,0	**1546**	1613	46,0	0,0
				848	2161	54,0	0,0	**91**	503	54,0	0,0
6	5	1	2	6289	718	73,0	0,0	**1350**	551	73,0	0,0
				7200	1062	76,9	19,0	**1184**	831	88,0	**0,0**
				7200	705	60,2	35,3	**1656**	540	80,0	**0,0**
				7200	792	53,5	30,5	**3293**	687	72,0	**0,0**
				7200	874	67,0	33,0	7200	969	68,0	**18,1**
6	6	1	2	7200	1068	66,0	**29,0**	7200	1374	66,0	**29,0**
				6792	2474	53,0	0,0	**2076**	3580	53,0	0,0
				521	2491	62,0	0,0	1149	3080	62,0	0,0
				7200	760	56,0	22,2	7200	86103	62,0	**1,6**
				7200	785	72,6	22,0	7200	1116	80,0	**14,0**

Cuadro 8.6.1.: Comparación de los algoritmos basados en LTC-LTX y LTP-LTX sobre las instancias de `Set2.2`, `Set2.3` y `Set2.4`. Parte I.

Algoritmo		LTC-LTX	LTP-LTX
# Variables		20647	32325
# Restricciones		11014	13135
# Óptimas	`Set2.2`	13	15
	`Set2.3`	6	12
	`Set2.4`	7	10
		26	37
# Factibles		19	8
Tiempo (s)	`Set2.2`	2388	976
	`Set2.3`	5398	3137
	`Set2.4`	5235	3352
		4340	2488
Tiempo opt. (s)		2250	1469
Nodos opt.		18296	17586
Gap (%)		7	2
Gap fact. (%)		16	12

Cuadro 8.6.2.: Comparación de los algoritmos basados en LTC-LTX y LTP-LTX sobre las instancias de `Set2.2`, `Set2.3` y `Set2.4`. Parte II.

un promedio de 12 % contra 19 %. En 9 de las 19 instancias donde LTC-LTX no prueba optimalidad, la cota inferior es al menos 5 % más baja que la reportada por LTP-LTX, y en 3 de estas 9, todas del `Set2.4`, la cota inferior llega a ser al menos 24 % más baja. En 8 de las 11 instancias en donde LTC-LTX no prueba optimalidad pero LTP-LTX sí, la mejor solución encontrada no es la óptima.

Con el objetivo de analizar los límites de estos algoritmos, se genera un nuevo conjunto de instancias con un gran número de pedidos y un horizonte de planificación más amplio.

- `Set2.5` considera el mapa de 3 localidades de la Figura 8.1.1a y se compone de dos instancias generadas aleatoriamente con $H = 40$ y $|P| = 40$ y dos con $H = 42$ y $|P| = 42$, todas con $|V| = 1$ y $|C| = 3$.

En estos casos, para encontrar soluciones iniciales heurísticas, se utiliza el mismo procedimiento que se describe al inicio de la Sección 8.4, con la salvedad de que la primera etapa se resuelve únicamente con la metaheurística LNS, debido al gran tamaño de las formulaciones de PLE.

Ambos algoritmos se evalúan sobre estas instancias, por medio de una computadora con mayor poder de cómputo, con las siguientes especificaciones: procesador Intel i7

2.9GHz con 16 threads, 6GB de memoria y sistema operativo Ubuntu 20.04 64bits. La ejecución de CPLEX se hace en modo paralelo, aprovechando la totalidad de los threads, con un límite de 10 horas para el tiempo de ejecución. Los demás parámetros del solver se mantienen con sus valores por defecto.

En lo que respecta a LTC-LTX, las instancias del `Set2.5` tienen un promedio de 158k variables y 92k restricciones. En el lapso de tiempo considerado, es posible resolver la relajación lineal del nodo raíz, para todas las instancias, y hasta llevar a cabo entre 3 y 5 rondas de planos de corte, las cuales reducen el gap relativo respecto a la solución inicial. Particularmente en una de ellas, las heurísticas de CPLEX logran encontrar una solución factible de mejor calidad que la inicial. Los gaps relativos reportados varían entre 6,7 % y 18,5 %. Sin embargo, vale la pena mencionar que no se llegan a realizar ramificaciones en ningún caso. En cambio para LTP-LTX, la memoria disponible se agota durante la escritura de la formulación de 3 de ellas, mientras que la restante llega a escribirse (485k variables y 186k restricciones), pero se agota la memoria durante el *presolve*. Por este motivo, no es posible computar ningún valor de gap relativo para esta formulación.

Finalmente, con el objetivo de revalidar la efectividad de los desarrollos realizados para LTC-LTX en esta segunda parte de la tesis, se decide ejecutar el algoritmo que resuelve su formulación base (ver Sección 6.4.2) sobre las instancias del `Set2.5`, es decir, sin restricciones alternativas ni desigualdades válidas. Para llevar esto a cabo, se usa la misma máquina y se mantienen los parámetros del solver detallados antes, además se sigue incorporando una solución inicial factible para garantizar la obtención de un valor del gap relativo a fin de compararlo con los anteriores. En este caso, el promedio de restricciones se reduce ligeramente a 87k. Se realiza un mayor número de rondas de planos de corte, entre 7 y 11, y al igual que antes, no se llegan a realizar ramificaciones en ningún caso. Lo más notable es el gran incremento que se observa en los valores de gaps relativos, variando entre 69,8 % y 79,4 %. Esto se atribuye al peor ajuste que tienen las relajaciones lineales en el nodo raíz y a la debilidad de los cortes de propósito general del solver, en comparación con las restricciones alternativas y las desigualdades válidas específicas encontradas en este trabajo.

8.7. Métodos secuenciales vs. simultáneos

Como última experiencia computacional, se propone comparar el comportamiento del mejor algoritmo exacto entre los métodos simultáneos, es decir, aquel basado en la formulación LTP-LTX, contra los algoritmos heurísticos presentados en los métodos secuenciales de la primera parte de la tesis. Para esta comparación se utilizan las instancias del `Set2.2`, `Set2.3` y `Set2.4`. Vale la pena aclarar que las instancias del `Set1.1` son demasiado grandes para los algoritmos exactos. Se conserva la misma computadora utilizada para estudiar los límites de los modelos simultáneos, cuyas especificaciones se encuentran en la última parte de la sección anterior.

Los desarrollos de la primera parte de la tesis se aplican de una forma parecida a lo propuesto para encontrar soluciones iniciales en la Sección 8.4. Para cada instancia, se realiza el siguiente procedimiento iterativo tantas veces como sea posible dentro de un tiempo límite dado de entrada. En cada iteración, se construyen a lo sumo 11 soluciones para la primera etapa de la descomposición secuencial. Una de ellas se obtiene por medio de la metaheurística LNS con los parámetros ya sugeridos por *irace*, salvo un límite de 10000 iteraciones y parámetros de ruido $ruido_{\text{ENTR}} = ruido_{\text{VIAJ}} = 0{,}1$. Las soluciones restantes, se obtienen resolviendo la formulación E1 con CPLEX, con todos sus parámetros por defecto, salvo modo de ejecución *oportunista* (no determinista) y *populate* con un gap relativo del 10 %. Estos valores fueron elegidos con la intención de diversificar la búsqueda. Luego, para cada una de estas soluciones de la primera etapa, se resuelve la segunda con la metaheurística GRASP×ILS con los parámetros ya sugeridos por *irace* y un límite de 200 iteraciones. Al finalizar, se retorna la mejor solución encontrada. Se proponen dos valores de tiempo límite para este estudio: 5 y 15 minutos. Además, en aquellas instancias donde la solución encontrada mejora al incrementar el tiempo límite, se realiza una tercera ejecución con un límite de tiempo de 2 horas.

Con respecto a los desarrollos de la segunda parte de la tesis, se aplica el algoritmo exacto basado en la formulación LTP-LTX, manteniendo la configuración final detallada en la sección anterior y considerando un límite de tiempo de 2 horas por instancia.

Los resultados obtenidos se muestran en el Cuadro 8.7.1. Las columnas 6-7 corresponden al método secuencial: "Valor" reporta el valor objetivo de la mejor solución encontrada dentro del límite de tiempo, "Gap (%)" es el porcentaje de gap relativo entre el valor objetivo de la mejor solución encontrada y la cota inferior reportada por CPLEX en el método simultáneo al finalizar su ejecución y "Tiempo (s)" es el menor límite de tiempo en segundos con el cual se encontró la mejor solución. Las columnas 8-10 corresponden al método simultáneo: "Cota sup." es el valor de la mejor solución factible encontrada, "Gap (%)" es el porcentaje de gap relativo reportado por CPLEX al finalizar la ejecución y "Tiempo (s)" es el tiempo de ejecución medido en segundos. Se resalta en negrita el gap relativo del método ganador y, en caso de empate, también se resalta el menor tiempo.

Comparando los métodos en base a la calidad de las soluciones encontradas, el simultáneo es claramente superior. En 34 de las 45 instancias consideradas, encuentra mejores soluciones, y en sólo una ocurre lo opuesto. Ambos encuentran soluciones de igual costo en otras 10 instancias, pero en 6 de ellas el simultáneo lo hace más rápido. El gap relativo promedio para el simultáneo es 2,1 % y para el secuencial es 7,8 %, es decir, hay una mejoría en los costos de aproximadamente 6 % en promedio. Para el método secuencial, el promedio de gap relativo con un límite de tiempo de 5 minutos es 8,5 % y baja a 8 % con un límite de tiempo de 15 minutos, mientras que al considerar 2 horas, se alcanza una mejor solución en una única instancia. Este último punto sugiere que al método secuencial no vale la pena ejecutarlo por más de 5 minutos para el tamaño de instancia considerado. También se destaca el número de instancias que el simultáneo resuelve por optimalidad, 40 contra 9. Por supuesto, la única forma de asegurar de que una solución encontrada secuencialmente es óptima es a partir de las cotas inferiores provistas por el otro método. Con respecto a los tiempos de ejecución, el secuencial es capaz de encontrar soluciones para todas las instancias limitando la ejecución a 5 minutos, mientras que simultáneo requiere 1563s en promedio, y en particular necesita más de 5 minutos en 24 instancias. Por supuesto, este estudio no escala con el tamaño de las instancias, ya que allí no sería posible aplicar los algoritmos exactos.

$\|L\|$	$\|R\|$	$\|V\|$	$\|C\|$	Secuencial			Simultáneo		
				Valor	Gap (%)	Tiempo (s)	Cota sup.	Gap (%)	Tiempo (s)
3	7	1	2	61	6,6	300	57	**0,0**	82
				87	13,8	300	75	**0,0**	3353
				73	6,8	300	68	**0,0**	44
				74	1,4	300	73	**0,0**	92
				70	0,0	300	70	**0,0**	**34**
3	8	1	2	105	14,1	300	92	**0,0**	121
				107	3,7	300	103	**0,0**	189
				93	17,2	900	77	**0,0**	52
				91	0,0	300	91	**0,0**	**20**
				89	7,9	300	82	**0,0**	23
3	9	1	2	76	9,2	300	69	**0,0**	41
				102	0,0	300	102	**0,0**	**116**
				117	12,8	300	102	**0,0**	448
				97	10,3	300	87	**0,0**	165
				132	0,8	300	131	**0,0**	83
3	8	2	4	64	14,1	300	55	**0,0**	484
				75	**0,0**	**300**	75	0,0	386
				101	**0,0**	**300**	101	0,0	1423
				83	7,2	300	77	**0,0**	505
				72	6,9	300	67	**0,0**	122
3	9	2	4	78	21,8	300	61	**0,0**	619
				122	14,8	900	120	**13,3**	7200
				75	6,7	300	70	**0,0**	16
				70	1,4	300	69	**0,0**	327
				89	2,2	300	87	**0,0**	1721
3	10	2	4	77	1,3	900	76	**0,0**	678
				80	11,2	300	71	**0,0**	1425
				100	1,0	7200	99	**0,0**	1384
				79	1,3	900	78	**0,0**	4932
				95	5,3	300	90	**0,0**	288
6	4	1	2	82	14,6	300	70	**0,0**	4764
				61	21,8	300	54	**11,6**	7200
				42	0,0	300	42	**0,0**	**91**
				53	13,2	300	46	**0,0**	479
				54	0,0	300	54	**0,0**	64
6	5	1	2	78	6,4	300	73	**0,0**	375
				88	0,0	300	88	**0,0**	**115**
				93	14,0	300	80	**0,0**	216
				76	5,3	300	72	**0,0**	4539
				74	**8,1**	900	83	18,1	7200
6	6	1	2	93	**29,0**	**300**	93	29,0	7200
				53	0,0	300	53	**0,0**	**215**
				64	3,1	900	62	**0,0**	493
				72	12,5	300	63	**0,0**	3832
				93	**22,5**	**900**	93	22,5	7200

Cuadro 8.7.1.: Comparación de los métodos secuenciales y simultáneos sobre las instancias de `Set2.2`, `Set2.3` y `Set2.4`.

9. Conclusiones

Durante esta segunda parte de la tesis, el problema se aborda desde un enfoque simultáneo, en el cual los dos subproblemas involucrados, ruteo de vehículos y planificación de tripulaciones, se programan al mismo tiempo. La ganancia de este enfoque es clave a la hora de garantizar la factibilidad del problema y de probar la optimalidad global de las soluciones encontradas.Esto supone grandes desafíos, debido a que deben construirse rutas separadas para los camiones y para los conductores, pero sincronizadas en tiempo y en espacio.

Se proponen seis formulaciones de PLE diferentes para el problema. Cada una de ellas utiliza dos digrafos auxiliares para representar el desplazamiento de los camiones y los conductores a lo largo del tiempo, respectivamente. Las rutas se modelan como caminos dirigidos en estos digrafos, pero algunos deben prohibirse puesto que no se corresponden con rutas válidas, siendo necesario incorporar nuevas familias de restricciones para evitarlos. A su vez, se presenta una novedosa alternativa para manejarlos, en la cual la estructura del digrafo se aumenta para que tales caminos no puedan existir por su construcción. Esto no es gratuito, digrafos más grandes implican más variables y restricciones en las formulaciones. En particular, el número de variables en los modelos que utilizan a G_{LT} y G_{LTC} para las rutas de camiones crece notoriamente con el número de camiones en comparación con G_{LTP}, y lo opuesto ocurre con el número de pedidos. Entre las bondades de los modelos, se destaca la simplicidad con la que pueden expresarse las restricciones de sincronización entre los camiones y los conductores. Todas las formulaciones propuestas son compactas, es decir, el número de variables y de restricciones crece polinomialmente con el tamaño de la instancia, a diferencia de la propuesta en [Bakarcic et al., 2013]. Por otro lado, fijando a 0 las variables relativas a los taxis, es posible resolver exactamente el mismo problema que ellos.

Mediante un amplio estudio de las relajaciones lineales de estas formulaciones, es posible identificar ciertas estructuras indeseables que ocurren con frecuencia en las soluciones fraccionarias. La existencia de las mismas tienen un impacto altamente desfavorable en el contexto de un algoritmo basado en branch-and-bound, dado que los valores óptimos fraccionarios pueden alejarse notoriamente de los valores óptimos enteros, brindando cotas duales poco ajustadas. Un gran número de familias de desigualdades válidas se propone para cortar tales soluciones y mitigar los inconvenientes mencionados. Algunas de ellas están compuestas por una cantidad de restricciones que crece exponencialmente con el tamaño de la instancia.

Se implementan algoritmos exactos basados en cada una de las formulaciones propuestas, utilizando como función objetivo una combinación lineal de los tres objetivos involucrados. En un principio, estos programas simplemente llaman a un solver de PLE de propósito general, pero gradualmente incorporan funcionalidades específicas más sofisticadas. Las experiencias computacionales llevadas a cabo exhiben comportamientos muy diferenciados según las formulaciones. A medida que los digrafos ganan estructura, y por ende las formulaciones necesitas menos familias de restricciones, el número de instancias resueltas por optimalidad aumenta de forma considerable, en parte debido al fortalecimiento de los valores óptimos de las relajaciones lineales. En base a estos resultados, se seleccionan dos de las formulaciones para continuar estudiando, ellas son LTC-LTX y LTP-LTX.

Por medio del intercambio de algunas restricciones de la formulación por otras alternativas, así como también de la incorporación de una solución heurística inicial construida con los desarrollos de la primera parte de la tesis, se obtienen importantes mejoras en la performance de los algoritmos, motivando la resolución de instancias un poco más grandes. Con respecto a la incorporación de desigualdades válidas, los resultados conseguidos son variados según la familia. En particular, las de viaje recolección-entrega resultan altamente efectivas en el algoritmo basado en LTC-LTX, ampliando notablemente el número de instancias resueltas. De hecho, estas desigualdades fortalecen tanto las relajaciones lineales que los valores óptimos fraccionarios llegan a ser muy similares a los de LTP-LTX, y en muchos casos son exactamente iguales. Por el otro lado, las de secuenciación de pedidos y de viaje en taxi tienen resultados un poco menos contundentes, y si bien algunas configuraciones permiten ampliar sutilmente el número de instancias resueltas y

reducir los tiempos de ejecución, otras configuraciones pueden llegar a empeorar el rendimiento de los mismos. Hay que mencionar que estas últimas desigualdades válidas no siempre son capaces de aumentar el valor óptimo de la relajación lineal, de hecho ocurre únicamente en pocas de las instancias consideradas, por lo que una posibilidad es que se precisen conjuntos de instancias específicas para terminar de discernir su efectividad.

Al comparar la mejor versión de cada uno de estos algoritmos, es claro que aquel que utiliza a LTP-LTX es superador, tanto en número de instancias resueltas como en tiempos de ejecución. A pesar de esto, es destacable la evolución en el rendimiento del algoritmo basado en LTC-LTX con cada uno de los desarrollos incorporados, logrando reducir notoriamente la brecha que existía originalmente entre ellos. Vistas las diferencias en las cotas inferior observadas entre ambos algoritmos en algunas de las instancias consideradas, todavía existe un margen para seguir mejorando al basado en LTC-LTX. Este resultado es particularmente importante en el camino hacia la resolución de instancias más grandes, dado que en general el número de pedidos crece más rápidamente que el número de camiones, lo que afecta profundamente al número de variables y de restricciones que posee LTP-LTX. Por lo tanto, es esperable que en instancias con pocos camiones y muchos pedidos, la única alternativa sea recurrir a LTC-LTX.

En conclusión, el estudio específico llevado a cabo en este trabajo de tesis es fundamental para el desarrollo de métodos efectivos que permitan dar soluciones de alta calidad. En particular, los propuestos en la primera parte de la tesis apuntan a resolver esta problemática de forma práctica, derivando en implementaciones capaces de dar respuestas rápidas a grandes instancias con hasta 3000 pedidos y un horizonte de planificación de hasta 28 días. Por su parte, la investigación realizada en la segunda parte es de carácter más teórico, confluyendo en algoritmos exactos capaces de encontrar soluciones con certificado de optimalidad en instancias no triviales, siendo ellas las primeras de su tipo en lo que a nosotros respecta para este problema. Otro punto importante que sugiere este trabajo, es que los métodos simultáneos pueden generar ahorros en los costos respecto a los secuenciales, de aproximadamente un 6 % en promedio. El trabajo realizado también da cuenta de lo complejo que puede ser la integración simultánea del ruteo de vehículos y planificación de tripulaciones y de los grandes desafíos que continúan abiertos a futuro, algunos de los cuales se mencionan a continuación.

Sería interesante revisar algunas de las simplificaciones realizadas en el enfoque secuencial, como por ejemplo considerar horarios de partida más flexibles para los taxis o incluso la posibilidad de que ellos transporten a más de un conductor a la vez (esto último también sería de interés en el enfoque simultáneo). Otra posible línea de trabajo involucra repensar la descomposición del problema, que actualmente ignora por completo a los conductores durante la etapa de ruteo de los camiones. Una posibilidad sería limitar la cantidad total de horas que se usan los camiones en todo intervalo de 24 horas, con el objetivo de que la posterior planificación de tripulaciones sea más sencilla. En relación a los algoritmos, queda pendiente estudiar la unificación de las heurísticas de reparación y de mejora, mediante la combinación de las funciones objetivo en una suma ponderada, cuyos pesos ideales deben ser determinados.

En lo que respecta al enfoque simultáneo, sería oportuno conocer más acerca del impacto que tienen cada una de las funciones objetivo en la dificultad del problema. Por ejemplo, sería interesante variar los pesos que se usan al combinar los objetivos para ponderar en mayor medida uno de ellos por sobre los otros. Por otro lado, CPLEX incorporó recientemente un solver multiobjetivo para optimización lexicográfica híbrida, el cual permite combinar en una suma ponderada aquellos objetivos que estén en un mismo nivel de jerarquía. A su vez, también sería valioso investigar otros abordajes provenientes de la optimización multiobjetivo.

En relación a los algoritmos exactos, sería deseable conocer más desigualdades válidas o incluso extender las ya conocidas a instancias con mayor número de camiones. Claro que esto supone un gran desafío, pero de encontrarlas sería posible que LTC-LTX saque ventaja sobre LTP-LTX por su imposibilidad para distinguir a los camiones. Otra posibilidad, es el desarrollo de algoritmos de separación específicos para algunas de las familias de desigualdades válidas, en especial para aquellas cuyo ajuste fue más notorio en el nodo raíz pero que están compuestas por demasiadas restricciones. En este sentido, es oportuno recordar que incluso para familias exponenciales podrían existir algoritmos de separación eficientes, aunque tales desarrollos en general son complejos y requieren de un profundo estudio específico del problema de separación asociado a cada familia. Por otro lado, los resultados obtenidos sugieren que sería altamente favorable contar con una heurística primal, visto lo difícil que puede resultar encontrar soluciones factibles durante la optimización.

La discretización del tiempo es otra gran arista para continuar con el estudio del problema. Por ejemplo, sería interesante comparar la calidad de las soluciones obtenidas para una misma instancia al variar la discretización del tiempo en intervalos de 30 min, 1 hora y 2 horas. En esta dirección, también sería de interés poder investigar modelos continuos que no asuman una discretización del tiempo.

Bibliografía

[Ait Haddadene et al., 2016] Ait Haddadene, S. R., Labadie, N., and Prodhon, C. (2016). A grasp × ils for the vehicle routing problem with time windows, synchronization and precedence constraints. *Expert Systems with Applications*, 66:274–294.

[Al Chami et al., 2018] Al Chami, Z., Manier, H., Manier, M.-A., and Chebib, E. (2018). An advanced grasp-hga combination to solve a multi-period pickup and delivery problem. *Expert Systems with Applications*, 105:262–272.

[Arabeyre et al., 1969] Arabeyre, J., Fearnley, J., Steiger, F., and Teather, W. (1969). The airline crew scheduling problem: A survey. *Transportation Science*, 3(2):140–163.

[Archetti and Savelsbergh, 2009] Archetti, C. and Savelsbergh, M. (2009). The trip scheduling problem. *Transportation Science*, 43(4):417–431.

[Bakarcic et al., 2013] Bakarcic, D., Piazza, G. D., Méndez-Díaz, I., and Zabala, P. (2013). An IP based heuristic algorithm for the vehicle and crew scheduling pick-up and delivery problem with time windows. In Cornelissen, K., Hoeksma, R., Hurink, J. L., and Manthey, B., editors, *12th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, Enschede, Netherlands, May 21-23, 2013*, volume WP 13-01 of *CTIT Workshop Proceedings*, pages 19–22.

[Baxter, 1981] Baxter, J. (1981). Local optima avoidance in depot location. *The Journal of the Operational Research Society*, 32(9):815–819.

[Carreto and Baker, 2002] Carreto, C. and Baker, B. (2002). A grasp interactive approach to the vehicle routing problem with backhauls. In *Essays and Surveys in Metaheuristics*, pages 185–199, Boston, MA. Springer US.

[Chvatal, 1983] Chvatal, V. (1983). *Linear Programming*. Series of books in the mathematical sciences. W. H. Freeman.

[Ciancio et al., 2018] Ciancio, C., Laganà, D., Musmanno, R., and Santoro, F. (2018). An integrated algorithm for shift scheduling problems for local public transport companies. *Omega*, 75:139–153.

[Cordeau et al., 2001] Cordeau, J.-F., Stojković, G., Soumis, F., and Desrosiers, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation science*, 35(4):375–388.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

[Dantzig and Ramser, 1959] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

[Delahaye et al., 2019] Delahaye, D., Chaimatanan, S., and Mongeau, M. (2019). Simulated annealing: From basics to applications. In *Handbook of Metaheuristics*, pages 1–35, Cham. Springer International Publishing.

[Diana and Dessouky, 2004] Diana, M. and Dessouky, M. M. (2004). A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological*, 38(6):539–557.

[DIMACS Challenges, 2002] DIMACS Challenges (2002). Instancias de coloreo de grafo. Recuperado de `https://mat.gsia.cmu.edu/COLOR04/`. Accedido el 26 de octubre de 2023.

[Domínguez-Martín et al., 2018] Domínguez-Martín, B., Rodriguez-Martin, I., and Salazar-Gonzalez, J.-J. (2018). The driver and vehicle routing problem. *Computers & Operations Research*, 92:56–64.

[Drexl, 2012] Drexl, M. (2012). Synchronization in Vehicle Routing - A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3):297–316.

[Drexl et al., 2013] Drexl, M., Rieck, J., Sigl, T., and Press, B. (2013). Simultaneous vehicle and crew routing and scheduling for partial- and full-load long-distance road transport. *Business Research*, 6(2):242–264.

[Dumas et al., 1991] Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22.

[Ehrgott, 2005] Ehrgott, M. (2005). *Multicriteria optimization*, volume 491. Springer Science & Business Media.

[Eveborn et al., 2006] Eveborn, P., Flisberg, P., and Rönnqvist, M. (2006). Laps care—an operational system for staff planning of home care. *European journal of operational research*, 171(3):962–976.

[Feo and Resende, 1989] Feo, T. A. and Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

[Feo and Resende, 1995] Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

[Fikar and Hirsch, 2017] Fikar, C. and Hirsch, P. (2017). Home health care routing and scheduling: A review. *Computers & Operations Research*, 77:86–95.

[Gendreau and Potvin, 2019] Gendreau, M. and Potvin, J.-Y. (2019). Tabu search. In *Handbook of Metaheuristics*, pages 37–55, Cham. Springer International Publishing.

[Gendreau et al., 2010] Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of metaheuristics*, volume 2. Springer.

[Goel, 2009] Goel, A. (2009). Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, 43(1):17–26.

[Goel and Irnich, 2017] Goel, A. and Irnich, S. (2017). An exact method for vehicle routing and truck driver scheduling problems. *Transportation Science*, 51(2):737–754.

[Goel and Kok, 2012] Goel, A. and Kok, L. (2012). Truck driver scheduling in the united states. *Transportation Science*, 46(3):317–326.

[Goel and Vidal, 2014] Goel, A. and Vidal, T. (2014). Hours of service regulations in road freight transport: An optimization-based international assessment. *Transportation Science*, 48(3):391–412.

[Goel et al., 2021] Goel, A., Vidal, T., and Kok, A. L. (2021). To team up or not: single versus team driving in European road freight transport. *Flexible Services and Manufacturing Journal*, 33(4):879–913.

[Haase et al., 2001] Haase, K., Desaulniers, G., and Desrosiers, J. (2001). Simultaneous vehicle and crew scheduling in urban mass transit systems. *Transportation science*, 35(3):286–303.

[Hansen et al., 2019] Hansen, P., Mladenović, N., Brimberg, J., and Pérez, J. A. M. (2019). Variable neighborhood search. In *Handbook of Metaheuristics*, pages 57–97, Cham. Springer International Publishing.

[Hansen et al., 2010] Hansen, P., Mladenović, N., and Moreno Pérez, J. A. (2010). Variable neighbourhood search: Methods and Applications. *Annals of Operations Research*, 175(1):367–407.

[Hansen and Mladenović, 2001] Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

[Hashimoto et al., 2008] Hashimoto, H., Yagiura, M., and Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456. In Memory of George B. Dantzig.

[Hemmelmayr et al., 2012] Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228.

[Ho and Szeto, 2016] Ho, S. C. and Szeto, W. (2016). Grasp with path relinking for the selective pickup and delivery problem. *Expert Systems with Applications*, 51(C):14–25.

[IBM, 2021] IBM (2021). ILOG CPLEX Optimization Studio 20.1.0. Recuperado de `https://www.ibm.com/products/ilog-cplex-optimization-studio`. Accedido el 20 de mayo de 2023.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

[Kleff, 2019] Kleff, A. (2019). *Scheduling and Routing of Truck Drivers Considering Regulations on Drivers' Working Hours*. PhD thesis, Karlsruher Institut für Technologie (KIT).

[Koç et al., 2018] Koç, Ç., Jabali, O., and Laporte, G. (2018). Long-haul vehicle routing and scheduling with idling options. *Journal of the Operational Research Society*, 69(2):235–246.

[Kok et al., 2010] Kok, A. L., Meyer, C. M., Kopfer, H., and Schutten, J. M. J. (2010). A dynamic programming heuristic for the vehicle routing problem with time windows and european community social legislation. *Transportation Science*, 44(4):442–454.

[Kontoravdis and Bard, 1995] Kontoravdis, G. and Bard, J. F. (1995). A grasp for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7(1):10–23.

[Lahyani et al., 2015] Lahyani, R., Khemakhem, M., and Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14.

[Lam et al., 2015] Lam, E., Van Hentenryck, P., and Kilby, P. (2015). Joint vehicle and crew routing and scheduling. In Pesant, G., editor, *Principles and Practice of Constraint Programming*, pages 654–670, Cham. Springer International Publishing.

[Laurent and Hao, 2009] Laurent, B. and Hao, J.-K. (2009). Iterated local search for the multiple depot vehicle scheduling problem. *Computers & Industrial Engineering*, 57(1):277–286.

[Lourenço et al., 2019] Lourenço, H. R., Martin, O. C., and Stützle, T. (2019). Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, pages 129–168, Cham. Springer International Publishing.

[López-Ibáñez et al., 2016] López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

[Ma et al., 2016] Ma, J., Ceder, A., Yang, Y., Liu, T., and Wei, G. (2016). A case study of beijing bus crew scheduling: A variable neighborhood-based approach. *Journal of Advanced Transportation*, 50:434–445.

[Martí et al., 2017] Martí, R., Panos, P., and Resende, M. (2017). *Handbook of Heuristics*. Handbook of Heuristics. Springer International Publishing.

[Mendes and Iori, 2020] Mendes, N. F. and Iori, M. (2020). A decision support system for a multi-trip vehicle routing problem with trucks and drivers scheduling. In *ICEIS (1)*, pages 339–349.

[Merz and Huhse, 2008] Merz, P. and Huhse, J. (2008). An iterated local search approach for finding provably good solutions for very large tsp instances. In *Parallel Problem Solving from Nature – PPSN X*, pages 929–939, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Mladenović and Hansen, 1997] Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

[Nasir and Kuo, 2020] Nasir, J. A. and Kuo, Y.-H. (2020). A decision support framework for home health care transportation with simultaneous multi-vehicle routing and staff scheduling synchronization. *Decision Support Systems*, 138:113361.

[Pisinger and Ropke, 2019] Pisinger, D. and Ropke, S. (2019). Large neighborhood search. In *Handbook of Metaheuristics*, pages 99–127, Cham. Springer International Publishing.

[Prescott-Gagnon et al., 2010] Prescott-Gagnon, E., Desaulniers, G., Drexl, M., and Rousseau, L.-M. (2010). European driver rules in vehicle routing with time windows. *Transportation Science*, 44(4):455–473.

[Rancourt et al., 2013] Rancourt, M.-E., Cordeau, J.-F., and Laporte, G. (2013). Long-haul vehicle routing and scheduling with working hour rules. *Transportation Science*, 47(1):81–107.

[Reeves, 2010] Reeves, C. R. (2010). Genetic algorithms. In *Handbook of Metaheuristics*, pages 109–139, Boston, MA. Springer US.

[Resende and Ribeiro, 2016] Resende, M. G. and Ribeiro, C. C. (2016). *Optimization by GRASP*. Springer New York.

[Resende and Ribeiro, 2019] Resende, M. G. C. and Ribeiro, C. C. (2019). Greedy randomized adaptive search procedures: Advances and extensions. In *Handbook of Metaheuristics*, pages 169–220, Cham. Springer International Publishing.

[Ropke and Pisinger, 2006] Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

[Salazar-González, 2014] Salazar-González, J.-J. (2014). Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. *Omega*, 43:71–82.

[Savelsbergh and Sol, 1998] Savelsbergh, M. and Sol, M. (1998). Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490.

[Schrijver et al., 2003] Schrijver, A. et al. (2003). *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer.

[Shaw, 1997] Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 46.

[Shaw, 1998] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer.

[Shen and Kwan, 2001] Shen, Y. and Kwan, R. (2001). Tabu Search for Driver Scheduling. *Computer-Aided Transit Scheduling*, 505:121–135.

[Solomon, 1987] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.*, 35:254–265.

[Tilk and Goel, 2020] Tilk, C. and Goel, A. (2020). Bidirectional labeling for solving vehicle routing and truck driver scheduling problems. *European Journal of Operational Research*, 283(1):108–124.

[Toth and Vigo, 2014] Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

[West et al., 2001] West, D. B. et al. (2001). *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

[Wolsey, 2020] Wolsey, L. A. (2020). *Integer programming*. John Wiley & Sons, second edition.

[Wolsey and Nemhauser, 1999] Wolsey, L. A. and Nemhauser, G. L. (1999). *Integer and combinatorial optimization*, volume 55. John Wiley & Sons.

[Xu et al., 2003] Xu, H., Chen, Z.-L., Rajagopal, S., and Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364.

Apéndices

1. Nociones de Grafos

Se introducen a continuación conceptos básicos y notación de teoría de grafos necesarios para la comprensión de la tesis. Se sugiere consultar a [West et al., 2001] para mayores detalles.

Un *digrafo* (o grafo dirigido) es un par ordenado $G = (N, E)$, donde N es un conjunto finito no vacío cuyos elementos se llaman *nodos* y E es un conjunto de pares ordenados de nodos distintos cuyos elementos se llaman *arcos*, i.e. $E \subseteq \{(u, v) \in N \times N : u \neq v\}$. Es usual notar al conjunto de nodos N de un digrafo G como $N(G)$ para evitar ambigüedades, y análogamente con el conjunto de arcos. Típicamente, los arcos tienen pesos asociados, dados por una función $w : E \to \mathbb{R}$, y a la tupla $G = (N, E, w)$ se la conoce como *digrafo ponderado*.

Un *subdigrafo* $G' = (N', E')$ de un digrafo $G = (N, E)$ es un digrafo que verifica $N' \subseteq N$ y $E' \subseteq E$.

Dado un arco $e = (u, v)$, se dice que e es incidente a u y v, que u es su *origen* y que v es su *destino*. Dos arcos son *adyacentes* si son incidentes a un mismo vértice y dos vértices son *adyacentes* si existe un arco que los tenga como origen y destino. Dos arcos son *paralelos* si tienen el mismo origen y el mismo destino. Un *multidigrafo* es equivalente a un digrafo, salvo que permite la existencia de arcos paralelos, es decir, E es un multiconjunto.

Un *camino dirigido* es una secuencia de nodos distintos tal que todo par ordenado de nodos consecutivos de la secuencia es un arco del digrafo, y se lo llama *ciclo dirigido* cuando los únicos nodos repetidos son el primero y el último. Si el primer

nodo del camino es u y el último es v, se dice que es un (u, v)-*camino dirigido*. Un digrafo es un *DAG*[1] si no tiene ciclos dirigidos.

Dado un nodo u, se denota con $\Gamma^-(u)$ al conjunto de *arcos de entrada*, i.e. $\Gamma^-(u) = \{(v, u) \in E : v \in N\}$, y con $\Gamma^+(u)$ al conjunto de *arcos de salida*, i.e. $\Gamma^+(u) = \{(u, v) \in E : v \in N\}$.

2. Nociones de Optimización Combinatoria y Complejidad Computacional

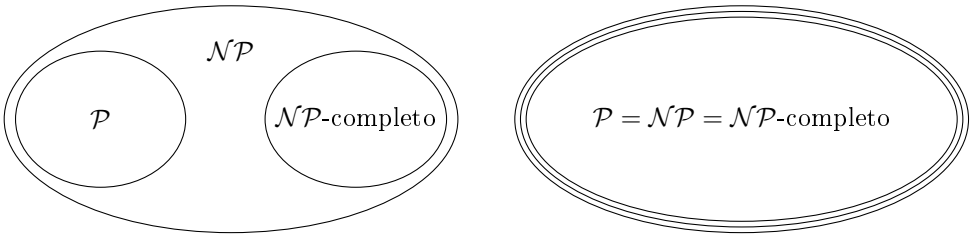
Se mencionan algunas nociones básicas de optimización combinatoria y de complejidad computacional. Una descripción más detallada puede encontrarse en [Wolsey and Nemhauser, 1999] y [Schrijver et al., 2003].

La *optimización combinatoria* se ocupa de buscar un objeto óptimo en una colección finita de objetos. Típicamente, la colección tiene una representación discreta, e.g. a través de un digrafo, y el número de objetos de la colección crece de forma exponencial en el tamaño de la representación.

La *complejidad computacional* se restringe a *problemas de decisión*, cuyas respuestas pueden ser "si" o "no". Esto no es una limitación, dado que todo problema de optimización, digamos $\text{mín}_{x \in X} f(x)$ siendo f una función a valores enteros, tiene su problema de decisión asociado: dado un número entero k, ¿existe $x \in X$ con $f(x) \leq k$?. Es sabido que a partir de un algoritmo de tiempo polinomial para el problema de decisión es posible derivar un algoritmo de tiempo polinomial para el problema de optimización.

Se nota con $\mathcal{P}$ a la colección de todos los problemas de decisión resolubles en tiempo polinomial; y con $\mathcal{NP}$ a la colección de todos los problemas de decisión para los cuales toda instancia con una respuesta positiva tiene un *certificado* verificable en tiempo polinomial de la correctitud de la respuesta. Trivialmente $\mathcal{P} \subseteq \mathcal{NP}$, pero existe una creencia generalizada de que estas clases son en realidad distintas, aunque aún no se ha podido demostrar. El problema de encontrar la relación entre

[1] Del inglés, Directed Acyclic Graph



(a) $\mathcal{P} \neq \mathcal{NP}$ (b) $\mathcal{P} = \mathcal{NP}$

Figura 2.1.: Clases de complejidad computacional.

estas clases, conocido como $\mathcal{P}$ vs. $\mathcal{NP}$, es considerado actualmente uno de los interrogantes abiertos más importantes en las ciencias de la computación.

Un problema en $\mathcal{NP}$ es $\mathcal{NP}$-*completo* si todo problema en $\mathcal{NP}$ puede *reducirse* a él, en tiempo polinomial. Por definición, son los más difíciles de $\mathcal{NP}$, dado que si un problema $\mathcal{NP}$-completo puede ser resuelto en tiempo polinomial, entonces puede derivarse un algoritmo de tiempo polinomial para todos los problemas en $\mathcal{NP}$. En la Figura 2.1, se muestran las relaciones entre estas clases de complejidad computacional según las posibles respuestas del problema $\mathcal{P}$ vs. $\mathcal{NP}$.

Se denomina $\mathcal{NP}$-*difícil* a la colección de problemas de optimización cuyos problemas de decisión asociados son $\mathcal{NP}$-completos. Muchos problemas de optimización combinatoria son $\mathcal{NP}$-difícil, lo que implica que no se conocen y no se esperan encontrar algoritmos capaces de resolver cualquier instancia en tiempo polinomial, pero esto no quiere decir que no valga la pena estudiarlos, sino todo lo contrario. En las últimas décadas, estudios específicos han permitido el desarrollo de algoritmos eficientes para muchos de estos problemas. Por supuesto, tales algoritmos siguen teniendo complejidad exponencial en el peor caso, pero pueden llegar a ser polinomiales en el caso promedio, e incluso para algunas familias de instancias con una cierta estructura particular, el problema restringido a estas entradas puede llegar ser resoluble en tiempo polinomial en el peor caso. Existen dos líneas de investigación para abordar este tipo de problemas: exactas y (meta)heurísticas. Dentro de las exactas, uno de los enfoques que ha resultado más exitoso es la Programación Lineal Entera.

3. Nociones de Programación Lineal Entera

Se presentan brevemente algunos fundamentos de Programación Lineal, Programación Lineal Entera, y esquemas algorítmicos para su resolución. Se sugiere ver [Chvatal, 1983] y [Wolsey, 2020] para una ampliación de los conceptos mencionados.

Un subconjunto $P \subseteq \mathbb{R}^n$ es un *poliedro* si se puede describir mediante un conjunto finito de desigualdades lineales, es decir, si existe una matriz A de dimensión $m \times n$ y un vector $b \in \mathbb{R}^m$ tal que $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Una desigualdad $c^T x \leq \delta$ es *válida* para P si es verificada por todo $x \in P$.

La *programación lineal (LP[2])* involucra el problema de minimizar o maximizar una función lineal sobre un poliedro, por ejemplo:

$$\text{mín}\{c^T x : Ax \leq b\}.$$

El poliedro P asociado al problema de LP se denomina *región factible* y cualquier vector en P es una *solución factible*. El problema es *factible* si $P \neq \emptyset$ y es *infactible* en caso contrario. La función $f(x) = c^T x$ es llamada *función objetivo*, y cualquier solución factible que alcance el *valor óptimo* es una *solución óptima*.

El método *simplex* de Dantzig es altamente eficiente en la práctica y en el caso promedio para resolver problemas de LP. Sin embargo, para casi todas las variantes del método se han encontrado familias ad hoc de problemas de LP que requieren un tiempo de ejecución exponencial. Por el contrario, los métodos del *elipsoide* de Khachiyan y de *punto interior* de Karmarkar tienen garantía de tiempo de ejecución polinomial en el peor caso. En particular, existen implementaciones para el método de Karmarkar bastante competitivas con simplex.

Un vector $x \in \mathbb{R}^n$ es *entero* si cada una de sus componentes es un entero, es decir, si $x \in \mathbb{Z}^n$. La *Programación Lineal Entera (PLE)* involucra el problema de minimizar o maximizar una función lineal sobre la intersección de un poliedro y el conjunto de vectores enteros, por ejemplo:

$$\text{mín}\{c^T x : Ax \leq b \wedge x \in \mathbb{Z}^n\}.$$

[2]Del inglés, linear programming.

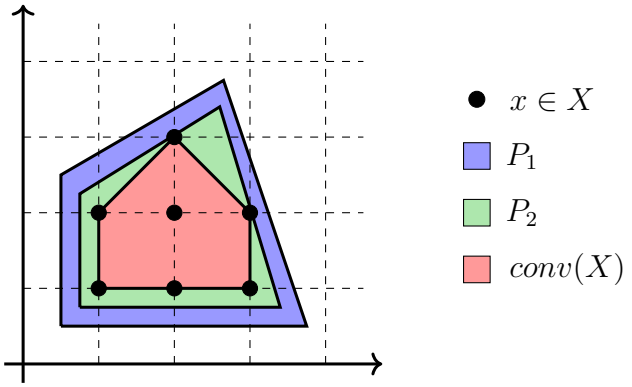


Figura 3.1.: Formulaciones para un conjunto $X \subseteq \mathbb{Z}^2$.

Las condiciones de integralidad de las variables hacen que el problema de resolver PLE sea $\mathcal{NP}$-difícil.

Dado un conjunto de soluciones $X \subseteq \mathbb{Z}^n$, un poliedro $P \subseteq R^n$ es una *formulación* para X si y solo si $X = P \cap \mathbb{Z}^n$. Dadas dos formulaciones P_1 y P_2 para X, se dice que P_2 es *más ajustada* que P_1 si $P_2 \subseteq P_1$. La formulación más ajustada para un conjunto X es la *cápsula convexa* de X, denotado $conv(X)$. Estos conceptos se ejemplifican en la Figura 3.1.

De hallarse una descripción completa de $conv(X)$, donde la cantidad de desigualdades lineales sea de tamaño polinomial en el número de variables, el problema de PLE podría resolverse mediante cualquier algoritmo de LP sobre $conv(X)$. Aún cuando la descripción de $conv(X)$ tenga alguna familia no polinomial de desigualdades lineales, el problema aún podría llegar a ser resuelto en tiempo polinomial si el *problema de separación* asociado a esa familia es resoluble en tiempo polinomial (dado un vector $x \in \mathbb{R}^n$, determinar si existe una desigualdad de la familia que sea violada por x). Es evidente que para problemas de optimización combinatoria $\mathcal{NP}$-difíciles, es de esperar que la descripción de $conv(X)$ tenga un número exponencial de desigualdades y que no se conozcan algoritmos de tiempo polinomial para su problema de separación, e incluso en muchos casos tampoco se cuenta con la descripción completa de $conv(X)$.

3.1. Esquemas algorítmicos para problemas de PLE

Existen diversos esquemas algorítmicos para resolver problemas de PLE, a continuación se mencionan los principales.

Branch-and-bound

El esquema de ramificación y poda, o *branch-and-bound*, consiste en un árbol de enumeración, donde los nodos representan subproblemas y las ramificaciones son particiones del espacio de búsqueda, que incorpora mecanismos de poda basados en cotas. Cada subproblema consiste en optimizar la función objetivo sobre los vectores enteros de un poliedro más restringido.

Existen tres criterios de poda: por infactibilidad (cuando el poliedro asociado al subproblema es vacío), por optimalidad (cuando se conoce la solución entera óptima del subproblema) y por cota (cuando existe un certificado de que una solución entera óptima del subproblema es peor que el *incumbente*, i.e. la mejor solución entera encontrada hasta el momento).

Una implementación de un algoritmo de tipo branch-and-bound involucra decidir: cómo dividir el problema sin perder soluciones enteras, en cuántos subproblemas ramificar cada nodo, cómo acotar el valor óptimo de los subproblemas, cómo recorrer el árbol, entre otras cosas.

Usualmente, el certificado consiste en resolver la *relajación lineal* del subproblema, es decir, resolver el problema de LP que se obtiene ignorando las condiciones de integralidad de las variables. En caso de que la solución óptima de la relajación lineal sea entera, entonces es una solución óptima del problema sin relajar, pudiendo así podar el nodo por optimalidad y quizás actualizar el incumbente. De lo contrario, es una cota inferior (o superior en caso de maximizar) y su comparación con el incumbente podría permitir podar el nodo por cota.

Planos de corte

El método de *planos de corte* consiste en el refinamiento iterativo del espacio de búsqueda mediante la introducción de desigualdades lineales, denominadas *cortes*, que preserven la región factible de soluciones enteras.

Típicamente, se comienza resolviendo la relajación lineal del problema. Si el problema es factible, el óptimo se encontrará en un punto extremo x^* del poliedro, pero puede no ser entero. En caso de no ser entero, se resuelve un problema de separación para encontrar un corte que separe a x^* de $conv(X)$ (cápsula convexa del conjunto de soluciones enteras). El corte se agrega a la formulación y se repite el procedimiento hasta que x^* sea entero.

Una implementación de un algoritmo de planos de corte depende en gran medida de la capacidad de generar cortes. Existen cortes de uso general que pueden aplicarse a cualquier problema de PLE, e.g. cortes Gomory. Sin embargo, los cortes que suelen ser más eficaces son los que representan *facetas*, es decir, que participan de la descripción minimal de $conv(X)$. Estos cortes son específicos de cada problema, e incluso su problema de separación puede ser $\mathcal{NP}$-difícil, debiendo recurrir a heurísticas que pueden no encontrar cortes aún cuando estos existan. En muchos casos, tampoco se espera conocer la descripción completa de $conv(X)$.

Otro uso de este método es cuando la cantidad de desigualdades lineales del problema de PLE es muy grande y deben ser manejadas bajo demanda.

Branch-and-cut

El método *branch-and-cut*, abreviado B&C, resulta de la combinación de branch-and-bound con planos de corte. Es probablemente el esquemas más popular. La idea es generar planos de corte en los nodos del árbol de enumeración para reforzar las formulaciones y favorecer la poda (por optimalidad y por cota), con el objetivo de reducir el número de nodos explorados.

En la implementación de un B&C, es necesario encontrar un balance en el énfasis puesto en cada uno de los esquemas. Generar una cantidad excesiva de cortes, además de consumir mucho tiempo, también hace más pesadas las formulaciones

pues incrementa el número de desigualdades lineales. Por este motivo, es usual limitar el número de cortes agregados en base a la profundidad del nodo del árbol.

Branch-and-price

El método *Branch-and-price*, abreviado B&P, es un branch-and-bound donde las relajaciones lineales asociadas a los nodos del árbol de enumeración se resuelven por medio de *generación de columnas*. Tiene su aplicación en problemas con una cantidad grande de variables (posiblemente exponencial), las cuales son gestionadas bajo demanda.

En cada nodo, se resuelve inicialmente un *problema maestro restringido* que tiene únicamente un subconjunto de las variables (columnas de la matriz). Para comprobar la optimalidad, se resuelve un *problema de pricing* para determinar si existen columnas no consideradas con el potencial de mejorar el valor objetivo. De no existir ninguna, entonces el problema maestro queda resuelto, de los contrario se incorporan las nuevas columnas al problema restringido y se vuelve a optimizar.

La aplicabilidad de B&P es bastante limitada por diversos motivos. Su implementación es tediosa y específica a cada problema. Requiere que la formulación tenga determinada estructura para poder aplicar generación de columnas y el problema de princing debe poder resolverse eficientemente. Las estrategias de ramificación clásicas son incompatibles al no preservar la estructura del problema, dificultando la aplicación del mismo algoritmo de princing en todos los nodos del árbol.

4. Nociones de Heurísticas y Metaheurísticas

Se introducen a continuación algunos conceptos preliminares de heurísticas y metaheurísticas. Para una ampliación de los mismos se sugiere consultar en [Gendreau et al., 2010, Resende and Ribeiro, 2016, Martí et al., 2017]. A su vez, en [Toth and Vigo, 2014] se abordan estos temas desde la perspectiva de problemas de ruteo de vehículos.

Las *heurísticas* se caracterizan por hallar soluciones *aceptables* en tiempos *razonables*, aunque no permiten certificar si las soluciones halladas son óptimas. Se

utilizan en general cuando las soluciones óptimas son poco prácticas, típicamente cuando el tamaño de las instancias de entrada provoca que los métodos exactos fallen o que los tiempos de cómputo no cumplan con los requerimientos prácticos.

Las heurísticas se suelen clasificar en dos grandes categorías: *heurísticas clásicas* y *metaheurísticas*. Las heurísticas clásicas se caracterizan por realizar una exploración relativamente limitada del espacio de búsqueda, y se clasifican a su vez en *heurísticas constructivas* y *heurísticas de mejora*. Por su parte, las metaheurísticas realizan una exploración mucho más robusta por las regiones más prometedoras del espacio de búsqueda, y suelen combinar las heurísticas clásicas con estrategias de alto nivel, estructuras de memoria, recombinación de soluciones, entre otros. Las soluciones metaheurísticas son usualmente de mejor calidad que las heurísticas, pero requieren de mayores tiempos de cómputo y de un tuneo de parámetros más riguroso.

4.1. Heurísticas constructivas

Las *heurísticas constructivas* comienzan con una solución vacía o incompleta e iterativamente la extienden hasta alcanzar una solución completa. Existen dos técnicas principales para la construcción de las soluciones, las cuales serán ejemplificadas en el contexto del ruteo de vehículos.

- *De inserción.* Los pedidos se insertan de a uno a la vez, hasta que todos hayan sido insertados. Las rutas se pueden construir de a una a la vez (*heurísticas de inserción secuenciales*) o de a varias (*heurísticas de inserción paralelas*). La elección de cuál pedido insertar y dónde insertarlo da lugar a diferentes variantes.

- *De ahorro.* Se comienzan construyendo rutas pequeñas y luego se van combinando una a una de acuerdo a un criterio de ahorro (¿qué ahorro se obtiene al combinar dos rutas?). La elección del criterio de ahorro y la forma en que se combinan las rutas da lugares a diferentes variantes.

Heurísticas constructivas de inserción

A continuación se mencionan algunas heurísticas constructivas de inserción recurrentes en la bibliografía.

- *De mejores.* Se elige el pedido que en ese momento tenga el menor *costo de inserción*, es decir, cuya inserción genere el menor incremento en el costo global, y se lo inserta en la mejor posición posible. Usualmente, se introduce azar en la elección para evitar construir sistemáticamente la misma solución, dando lugar a una heurística *randomizada avara* o *semigreedy*.

  Una manera clásica de randomizar la inserción es eligiendo el pedido al azar de una *lista de candidatos restringida (LCR)*. Suele estar limitada por un *parámetro de calidad* $\alpha \in [0, 1]$, de modo que un pedido p pertenece a la LCR si su inserción es factible y su costo de inserción ci_p verifica

  $$ci_{min} \leq ci_p \leq ci_{min} + \alpha.(ci_{max} - ci_{min}),$$

  donde ci_{min} y ci_{max} son el menor y el mayor costo de inserción entre todos los pedidos. En particular, la inserción resulta puramente aleatoria cuando $\alpha = 1$ y puramente greedy cuando $\alpha = 0$.

  Otra alternativa consiste en agregar un término de ruido en la función objetivo. Por ejemplo, cada vez que se calcula el costo de insertar un pedido p en algún lugar de una ruta, se suma un número de ruido generado al azar en un cierto intervalo.

- *De arrepentimiento.* Se elige el pedido que en ese momento genere el mayor arrepentimiento en caso de no poder insertarlo en su mejor ruta y haya que insertarlo en su k-ésima mejor ruta, donde k es un parámetro que indica cuán a futuro mirar.

  En primer lugar, se eligen los pedidos que no puedan insertarse en al menos k rutas, y luego se elige el pedido p que maximice el *valor de arrepentimiento* dado por

  $$\sum_{j=1}^{k} (ci_p^j - ci_p^1),$$

siendo ci_p^k es el costo de insertar el pedido p en la k-ésima mejor ruta. El pedido elegido se inserta en su mejor posición posible.

4.2. Heurísticas de mejora

Las *heurísticas de mejora* se ocupan de modificar una solución inicial mediante la aplicación sistemáticamente de un conjunto preseleccionado de operaciones, con el objetivo de producir una nueva y mejor solución, típicamente con un mejor valor objetivo. Si la modificación conduce a una mejor solución, entonces la solución actual es remplazada por la nueva solución y el procedimiento se repite.

Siendo $\mathcal{S}$ el conjunto de soluciones factibles, una *estructura de vecindario* es una función $N : \mathcal{S} \to \mathcal{P}(\mathcal{S})$. El subconjunto de soluciones $N(sol)$ es el *vecindario* de una solución sol y cada solución sol' de $N(sol)$ es un *vecino* de sol. Siendo $w : \mathcal{S} \to \mathbb{Z}$ la función objetivo a minimizar, una solución sol es *localmente óptima*, o un *óptimo local*, con respecto a N si $w(sol) \leq w(sol')$ para todo $sol' \in N(sol)$.

En el Algoritmo 11, se describe el pseudocódigo de una *búsqueda local*. La entrada consiste en una solución sol y una estructura de vecindario N. Se mantienen tres variables: sol^* es la solución actual (que es también la mejor solución encontrada hasta el momento), sol' es el mejor vecino de la solución actual y *optLocal* es una bandera que indica si la solución actual es un óptimo local con respecto a N. En las líneas 1–2, se inicializan la solución actual y la bandera de optimalidad local. En la línea 3, se controla si se cumple el criterio de parada, que en este caso consiste en alcanzar un óptimo local. En la línea 4, se busca un vecino de la solución actual que tenga el menor valor objetivo. En la líneas 5, se evalúa si el valor objetivo del vecino elegido es menor al de la solución actual. De ser así, en la línea 6, se actualiza la solución actual, y de lo contrario, en la línea 8, se enciende la bandera de optimalidad local. La búsqueda del vecino de menor costo generalmente requiere la exploración exhaustiva del vecindario, pero también es habitual detener anticipadamente la exploración al hallar al menos un vecino que mejore el valor objetivo. En la línea 9, se retorna la mejor solución encontrada.

Usualmente, los vecindarios se definen a partir de *movimientos* que son operaciones que transforman una solución factible sol en otra solución sol', posiblemente infactible, de forma que ambas soluciones comparten parte de su estruc-

Algoritmo 11 `búsquedaLocal`

Entrada: $sol \in \mathcal{S}$ solución inicial, $N : \mathcal{S} \to \mathcal{P}(\mathcal{S})$ estructura de vecindario.
Salida: $sol^* \in \mathcal{S}$ óptimo local.

```
1: sol* ← sol
2: optLocal ← false
3: while not optLocal do
4:     sol' ← argmin{w(sol') : sol' ∈ N(sol*)}
5:     if w(sol') < w(sol*) then
6:         sol* ← sol'
7:     else
8:         optLocal ← true
9: return sol*
```

tura. El movimiento m define un *vecindario extendido* $N_m^{\text{INF}}(sol)$ compuesto por todas las soluciones factibles e infactibles que se obtienen de aplicar m a sol de todas las formas posibles y también define un *vecindario de soluciones factibles* $N_m(sol) = N_m^{\text{INF}}(sol) \cap \mathcal{S}$.

Los movimientos aplicados en VRPs se clasifican en dos categorías principales.

- *De única ruta.* Aplican cambios sobre una única ruta. Por ejemplo, permutar algunos pedidos de la ruta, modificar el horario programado para un pedido, entre otros.
- *De múltiples rutas.* Aplican cambios entre dos o más rutas. Por ejemplo, intercambiar o mover pedidos entre las rutas.

Una mejora simple pero eficaz cuando la búsqueda queda atascada en un óptimo local, consiste en saltar a una estructura de vecindario diferente. Esta mejora se menciona a continuación.

Variable Neighborhood Descent

En un *Variable Neighborhood Descent (VND)*, se consideran múltiples estructuras de vecindario N_i con $i = 1, \ldots, k$, para algún número natural k. Una posibilidad (existen otras) para explorar secuencialmente estas estructuras de vecindario es la siguiente. La idea es realizar una búsqueda local sobre la i-ésima estructura de vecindario, inicialmente con $i = 1$. Tras alcanzar un óptimo local con respecto a N_i, se cambia a la estructura de vecindario N_{i+1} y se inicia una nueva búsqueda

local. Este procedimiento se repite hasta alcanzar un óptimo local con respecto a todas las estructuras de vecindario, que llamaremos un *óptimo local general*.

En el Algoritmo 12, se describe el pseudocódigo de VND. La entrada es una solución inicial sol y una estructura de vecindario N_i para cada $i = 1, \ldots, k$. Se mantienen 4 variables: sol^* es la solución actual (que es también la mejor solución encontrada hasta el momento), $optLocal$ es una bandera que indica si la solución actual es un óptimo local general, i es el índice de la estructura de vecindario que se está explorando en la iteración actual y sol' es un óptimo local con respecto al i-ésimo vecindario. En las líneas 1–2, se inicializan la solución actual y la bandera. En la línea 3, se evalúa si se alcanzó el criterio de parada, que en este caso consiste en encontrar un óptimo local general. En la línea 4, se prende la bandera para indicar que se presume que la solución actual es un óptimo local general. En la línea 5, se marca al primer vecindario para explorar. En la línea 6, se controla si se han explorado todos los vecindarios. En la línea 7, se aplica una búsqueda local a la solución actual usando la i-ésima estructura de vecindario, almacenando el óptimo local resultante en sol'. En la línea 8, se evalúa si el costo del óptimo local es mejor al de la solución actual. De ser así, en la línea 9, se actualiza la mejor solución y, en la línea 10, se desactiva la bandera dado que se desconoce si la solución actual es un óptimo local para los vecindarios previamente considerados. En la línea 11, se pasa a la siguiente estructura de vecindario. En la línea 12, se retorna la mejor solución encontrada.

Algoritmo 12 `VND`

Entrada: $sol \in \mathcal{S}$ solución inicial, $N_i : \mathcal{S} \to \mathcal{P}(\mathcal{S})$ estructura de vecindario con $i = 1, \ldots, k$.
Salida: $sol^* \in \mathcal{S}$ óptimo local con respecto a N_i para todo $i = 1, \ldots, k$.

```
 1: sol* ← sol
 2: optLocal ← false
 3: while not optLocal do
 4:     optLocal ← true
 5:     i ← 1
 6:     while i ≤ k do
 7:         sol' ← búsquedaLocal(sol*, N_i)
 8:         if w(sol') < w(sol*) then
 9:             sol* ← sol'
10:             optLocal ← false
11:         i ← i + 1
12: return sol*
```

Una implementación de un VND requiere decidir cuántas estructuras de vecindario usar y en qué orden explorarlas. Usualmente, las estructuras de vecindario menos complejas son las primeras en explorarse.

4.3. Metaheurísticas

Esta sección se limita a presentar los esquemas básicos de las metaheurísticas usadas en los desarrollos de la tesis.

Large Neighborhood Search

La metaheurística *Large Neighborhood Search (LNS)*, propuesta en [Shaw, 1998], sistemáticamente aplica un método de destrucción para destruir parte de la solución actual y un método de reparación para volver a construir una solución, posiblemente distinta a la destruida, de forma tal que diferentes partes de la solución sean destruidas en cada iteración. El vecindario de una solución queda definido implícitamente como el conjunto de soluciones que se pueden obtener aplicándole a la solución primero el método de destrucción y luego el de reparación.

En el Algoritmo 13, se muestra el pseudocódigo de la metaheurística LNS. La variable sol representa a la solución actual, sol^* a la mejor solución encontrada hasta el momento y sol' a una solución temporal. En la línea 1, se inicializa la mejor solución. En la línea 2, se controla si se cumple el criterio de parada, usualmente se limita en el número de iteraciones o se establece un tiempo límite de ejecución. En la línea 3, la solución actual es destruida y luego reparada para obtener una nueva solución sol'. La función `destruir` retorna una copia parcialmente destruida de la solución, mientras que `reparar` retorna una solución factible construida a partir de la solución destruida. En las líneas 4–5, se verifica si se cumple el criterio de aceptación para la solución recién reparada, y de ser así se convierte en la nueva solución actual. La función `aceptar` se puede implementar de diferentes formas, lo más sencillo es aceptar únicamente soluciones con mejor valor objetivo. En las líneas 6–7, se evalúa si el costo de la nueva solución es mejor que el de la mejor solución encontrada, y de ser necesario se actualiza la mejor solución. En la línea 8, se retorna la mejor solución encontrada.

Algoritmo 13 LNS

Entrada: sol : solución factible.
Salida: sol^* : solución factible.

```
1: sol* ← sol
2: while no se cumple criterio de parada do
3:     sol' ← reparar(destruir(sol))
4:     if aceptar(sol', sol) then
5:         sol ← sol'
6:     if w(sol') < w(sol*) then
7:         sol* ← sol'
8: return sol*
```

Una de las decisiones más importantes cuando se implementa un LNS, es el *grado de destrucción*. Destruir sistemáticamente una parte muy pequeña de la solución restringirá demasiado la exploración del espacio de búsqueda. Por el contrario, destruir una parte muy grande tampoco es conveniente dado que degrada a LNS a múltiples construcciones comenzando con soluciones prácticamente vacías. Esto puede consumir demasiado tiempo si la reparación construye soluciones óptimas (por ejemplo con branch-and-bound, o algún otro método exacto) o puede llevar sistemáticamente a soluciones de pobre calidad si la reparación es semigreedy.

Simulated Annealing

La metaheurística *Simulated Annealing (SA)*, formalizada en [Kirkpatrick et al., 1983], consiste en una búsqueda local con la capacidad de escapar de óptimos locales. En cada iteración de SA, la solución actual es comparada con alguno de sus vecinos. Las soluciones que mejoran el valor objetivo siempre son aceptadas, mientras que las que empeoran son aceptadas con una cierta probabilidad. La probabilidad de aceptar peores soluciones depende de un parámetro de temperatura, generalmente decreciente en el número de iteraciones, que hace que los movimientos a estas soluciones ocurran con menor frecuencia a medida que la temperatura se acerca a cero.

Esta metaheurística tiene su analogía en un fenómeno termodinámico, en el cual un sólido cristalino se calienta y luego se deja enfriar lentamente hasta que alcanza su estructura más regular, y por lo tanto libre de defectos cristalinos. Si el régimen de

enfriamiento es lo suficientemente lento, el resultado es un sólido con características estructurales superiores.

El pseudocódigo de la metaheurística SA se muestra en el Algoritmo 14. La entrada consiste en una estructura de vecindario N, un parámetro de temperatura $t_k > 0$ decreciente para cada $k = 0, \ldots, k_{max}$, donde k_{max} es un límite en el número de pasos, y un límite en el número de iteraciones IT_k para cada temperatura t_k. Se mantienen las siguientes variables: s es la solución actual, k indica el número de iteraciones realizadas por el bucle exterior, it_k indica el número de iteraciones realizadas por el bucle interior, s' es un vecino de la solución actual y $\Delta_{s,s'}$ es la diferencia de valor objetivo entre s' y s. En la línea 1, se genera una solución factible inicial. En las líneas 2 y 4, se inicializan los contadores de iteraciones. En las líneas 3 y 5, se controla si se cumplen los criterios de parada, que en este caso es un límite en el número de iteraciones. En la línea 6, se selecciona un vecino de la solución actual, puede ser de forma aleatoria o con alguna regla predefinida. En la línea 7, se calcula la diferencia de valor objetivo entre ambas soluciones. En las líneas 8–9, se actualiza la solución actual en caso de que haya una mejora en el valor objetivo. Por el contrario, si la función objetivo empeora, se actualiza la solución actual con una probabilidad dada por una función exponencial, en las líneas 10–11. En las líneas 12 y 13, se aumentan los contadores de iteraciones. En la línea 14, se retorna la solución actual.

Una de las decisiones más importantes al implementar SA es el régimen de enfriamiento, el cual afecta la velocidad de convergencia. El régimen de enfriamiento queda completamente definido por una temperatura inicial, una tasa de enfriamiento (¿cuánto decrece la temperatura en cada iteración?) y un número máximo de iteraciones. Típicamente, la temperatura inicial se configura de forma tal que la probabilidad de aceptar soluciones peores a la inicial sea igual a determinado valor.

Greedy Randomized Adaptative Search Procedure

La metaheurística *Greedy Randomized Adaptative Search Procedure (GRASP)* [Feo and Resende, 1989, Feo and Resende, 1995, Resende and Ribeiro, 2016], consiste en múltiples iteraciones independientes, cada una compuesta por una fase de

Algoritmo 14 Simulated Annealing

Entrada: N : estructura de vecindario, $t_k > 0$ para cada $k = 0, \ldots, k_{max}$, $IT_k > 0$ para cada $k = 0, \ldots, k_{max}$.
Salida: s : solución factible.

```
1: generar solución factible s
2: k ← 0
3: while k ≤ k_max do
4:     it_k ← 0
5:     while it_k < IT_k do
6:         generar s' ∈ N(sol)
7:         Δ_{s,s'} ← w(s') − w(s)
8:         if Δ_{s,s'} ≤ 0 then
9:             s ← s'
10:        else
11:            s ← s' con probabilidad e^{−Δ_{s,s'}/t_k}
12:        it_k ← it_k + 1
13:    k ← k + 1
14: return s
```

construcción y una posterior fase de mejora. Durante la fase de construcción, se construye una solución factible inicial usando alguna heurística constructiva de inserción, la cual no debe ser puramente avara, sino estar randomizada, por ejemplo mediante una lista de candidatos restringida (LCR). En general, esta heurística también es *adaptativa*, en el sentido de que los costos de inserción pueden no ser constantes en cada iteración, sino depender de las decisiones tomadas previamente. En caso de que la solución construida no sea factible, es posible aplicar algún procedimiento de reparación. La fase de mejora consiste en aplicar una heurística de mejora, lo más simple es una búsqueda local que explore una estructura de vecindario hasta alcanzar un óptimo local. Mientras tanto, la mejor solución encontrada es almacenada y retornada al final de la ejecución.

El pseudocódigo de GRASP se presenta en el Algoritmo 15. La entrada es un parámetro de calidad α para limitar el tamaño de la LCR en la fase de construcción. Se mantienen dos variables: *sol* es una solución temporal y sol^* es la mejor solución factible encontrada hasta el momento. En la línea 1, se inicializa el costo de la mejor solución encontrada en infinito. En la línea 2, se controla si se cumple el criterio de parada, típicamente un número máximo de iteraciones o un tiempo límite de ejecución. En la línea 3, se construye una solución inicial *sol* mediante una heurística constructiva de inserción randomizada por α. En las líneas 4–5, se

aplica un procedimiento de reparación para factibilizar la solución de ser necesario. En la línea 6, se aplica una búsqueda local y se almacena el resultado en *sol*. En las líneas 7–9, se actualiza la mejor solución de ser necesario. En la línea 10, se retorna la mejor solución.

Algoritmo 15 GRASP

Entrada: $\alpha \in [0, 1]$.
Salida: sol^* : solución factible.

```
1:  w(sol*) ← ∞
2:  while no se cumple criterio de parada do
3:      sol ← construirRand(α)
4:      if sol no es factible then
5:          sol ← reparar(sol)
6:      sol ← búsquedaLocal(sol)
7:      if w(sol) < w(sol*) then
8:          sol* ← sol
9:          w(sol*) ← w(sol)
10: return sol*
```

Una de las decisiones más importantes en la implementación de GRASP es el valor del parámetro α. Pensando que en cada iteración se muestrea una solución desde una distribución desconocida, entonces el valor objetivo promedio y la varianza dependen de α. Por ejemplo, cuando $\alpha = 0$ (puramente aleatorio), se espera que el valor objetivo promedio esté lejos del valor óptimo y que la varianza sea grande; por el contrario cuando $\alpha = 1$ (puramente greedy), siempre se genera la misma solución por lo que la varianza es 0. Ninguno de estos valores extremos suele funcionar bien en la práctica. En general, las mejores soluciones se pueden obtener cuando el valor objetivo promedio es relativamente bajo en presencia de varianza relativamente alta. Otro aspecto que influye es el tiempo requerido por la búsqueda local, el cual tiende a ser mayor cuanto más aleatoria es la construcción.

Iterated Local Search

La metaheurística *Iterated Local Search (ILS)* se caracteriza por explorar el espacio de búsqueda siguiendo una única cadena de soluciones generadas iterativamente por medio de una heurística embebida, típicamente una búsqueda local. La próxima solución de la cadena surge de aplicar la búsqueda local sobre una solución

intermedia, la cual presenta un cambio o *perturbación* respecto a la mejor solución encontrada hasta el momento. Se espera que la perturbación produzca una solución cercana (de acuerdo a alguna métrica) para evitar saltos muy grandes en el espacio de búsqueda, pero lo suficientemente distinta para escapar de la *base de atracción* (conjunto de soluciones para las cuales la búsqueda local retorna la misma solución).

El pseudocódigo de la metaheurística ILS se describe en el Algoritmo 16. Se mantienen 3 variables: sol es la solución actual, sol^* es la mejor solución encontrada hasta el momento e *historia* es una memoria que puede ser explotada para tomar decisiones más efectivas a partir del comportamiento en las iteraciones pasadas. En la línea 1, se genera una solución inicial, que puede ser aleatoria o generada con alguna heurística constructiva, y en la línea 2, se le aplica la búsqueda local. En la línea 3, se inicializa la mejor solución encontrada con la solución actual. En la línea 4, se comprueba si se cumple el criterio de parada, típicamente un límite en el número de iteraciones o en el tiempo de cómputo. En la línea 5, se genera una solución intermedia aplicando una perturbación sobre la mejor solución, y en la línea 6, se le aplica la búsqueda local. En las líneas 7–8, se evalúa si el óptimo local retornado cumple el criterio de aceptación, y de ser necesario se actualiza la mejor solución. Lo más simple es aceptar únicamente soluciones que mejoren en valor objetivo. En la línea 9, se retorna la mejor solución.

Algoritmo 16 ILS

Salida: sol^* : solución factible
1: $sol \leftarrow$ `construir`()
2: $sol \leftarrow$ `búsquedaLocal`(sol)
3: $sol^* \leftarrow sol$
4: **while** no se cumple criterio de parada **do**
5: $sol \leftarrow$ `perturbar`($sol^*, historia$)
6: $sol \leftarrow$ `búsquedaLocal`(sol)
7: **if** `aceptar`($sol^*, sol, historia$) **then**
8: $sol^* \leftarrow sol$
9: **return** sol^*

En una implementación de ILS es fundamental decidir en cuánto la perturbación cambiará a la solución. Si la perturbación es muy fuerte, la metaheurística hará en cada iteración un reinicio a una solución aleatoria, y será poco probable encontrar buenas soluciones. Por el contrario si es muy débil, la búsqueda local caerá con

frecuencia en óptimos locales ya visitados, lo que limita la diversificación de la exploración. Comúnmente, la perturbación aplica aleatoriamente un movimiento predefinido, el cual no es utilizado por la búsqueda local (para dificultar que el cambio pueda ser revertido). Perturbaciones más complejas pueden utilizar la memoria para no caer en soluciones ya visitadas o para ajustar de forma adaptativa la fuerza de la perturbación.

Híbridas

En el los últimos años ha habido un auge de las llamadas *metaheurísticas híbridas*, que combinan componentes de diferentes metaheurísticas para explorar de forma más efectiva el espacio de búsqueda. En particular, en esta tesis se aplicarán metaheurísticas híbrida que combina LNS con SA y GRASP con ILS.

También vale la pena mencionar que se han vuelto cada vez más populares los híbridos de metaheurísticas con otras técnicas de optimización, tales como branch-and-bound o *constraint programming*.

5. Optimización multiobjetivo

Por último, se introducen algunos conceptos elementales sobre optimización multiobjetivo, necesarios para comprender algunos de los desarrollos realizados en la tesis. Para una ampliación de estos temas, se recomienda consultar en [Ehrgott, 2005].

Un problema de optimización multiobjetivo es de la forma

$$\begin{aligned} \text{mín} \quad & (f_1(x), \ldots, f_p(x)) \\ \text{sujeto a} \quad & x \in \mathcal{X}, \end{aligned}$$

donde $\mathcal{X}$ es el *conjunto factible* y $f = (f_1, \ldots, f_p) : \mathcal{X} \to \mathbb{R}^p$ es el vector de funciones objetivo a minimizar. A diferencia de la optimización clásica ($p = 1$), es posible asociar diferentes interpretaciones al operador de minimización.

En la interpretación más difundida, se buscan soluciones factibles para las cuales no sea posible mejorar ningún objetivo sin empeorar otro. Formalmente, una solución factible $\hat{x} \in \mathcal{X}$ se dice *eficiente* o *Pareto optimal*, si no existe $x \in \mathcal{X}$ tal que $f_k(x) \leq f_k(\hat{x})$ para $k = 1, \ldots, p$ y $f_i(x) < f_i(\hat{x})$ para algún $i \in \{1, \ldots, p\}$. Si $\hat{x}$ es Pareto optimal, entonces $f(\hat{x})$ se llama *punto no dominado*. El conjunto de todas las soluciones Pareto optimales $\hat{x} \in \mathcal{X}$ se llama *conjunto eficiente* y se denota $\mathcal{X}_E$. El conjunto de todos los puntos no dominados $\hat{y} = f(\hat{x}) \in \mathcal{Y}$, con $\hat{x} \in \mathcal{X}_E$, se llama *conjunto no dominado* o *frontera de Pareto* y se denota $\mathcal{Y}_N$. Por lo tanto, resolver un problema de optimización multiobjetivo significa determinar el conjunto eficiente $\mathcal{X}_E$.

Existen dos puntos de particular interés a la hora de acotar los rangos de valores que pueden alcanzar los puntos no dominados. Siendo los puntos $y^I = (y_1^I, \ldots, y_p^I)$ e $y^N = (y_1^N, \ldots, y_p^N)$ definidos por

$$y_k^I := \min_{x \in \mathcal{X}} f_k(x) \text{ para } k = 1, \ldots, p,$$
$$y_k^N := \max_{x \in \mathcal{X}_E} f_k(x) \text{ para } k = 1, \ldots, p,$$

es claro que $y_k^I \leq y_k \leq y_k^N$ para todo $k = 1, \ldots, p$ e $y \in \mathcal{Y}_N$. En particular, y^I es llamado *punto ideal* e y^N *punto nadir*. Notar que el cálculo de y^N considera únicamente las soluciones factibles de $\mathcal{X}_E$, en lugar de $\mathcal{X}$, debido a que el valor máximo de las funciones objetivo podría ser demasiado grande (e incluso no acotado) en las soluciones de $\mathcal{X} \setminus \mathcal{X}_E$. En general ocurre que $y^I \notin \mathcal{Y}_N$, dado que los objetivos suelen estar en conflicto, es decir, incrementar el valor de uno de ellos suele afectar a otro, y por lo tanto no existe solución factible que pueda minimizar al mismo tiempo a cada función objetivo individual.

En la Figura 5.1 se presenta un ejemplo en $\mathbb{R}^2$, en el cual se minimiza un vector de funciones objetivo $f = (f_1, f_2)$. La gráfica muestra el valor de f_1 en función de f_2 para las soluciones factibles (región celeste). También, se ilustra la frontera de Pareto (línea azul), los puntos ideal y nadir y dos puntos no dominados $\hat{y}^1$ y $\hat{y}^2$.

Tradicionalmente, un problema de optimización multiobjetivo se resuelve por medio de *escalarización*, es decir, convirtiéndolo en un problema de optimización con un único objetivo, preservando ciertas propiedades del original.

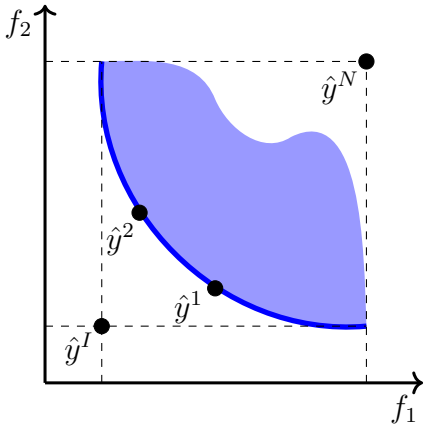


Figura 5.1.: Ejemplo de frontera de Pareto en $\mathbb{R}^2$.

El método de escalarización más simple consiste en combinar las funciones objetivo con una suma ponderada, es decir, resolver el problema de optimización

$$\begin{aligned} \text{mín} \quad & \sum_{k=1}^{p} \lambda_k . f_k(x) \\ \text{sujeto a} \quad & x \in \mathcal{X} \end{aligned}$$

donde λ es un vector de $\mathbb{R}^p_{\geq 0}$, que representa el *peso* o la *importancia* de cada objetivo sobre el valor final. En particular, se puede probar que si $\lambda \in \mathbb{R}^p_{>0}$, entonces toda solución óptima de este último problema es una solución Pareto optimal del original. Es esperable que una gran diversidad de soluciones se obtenga al resolver este problema variando λ, pero pueden aparecer algunos inconvenientes. En particular, si los objetivos tienen rangos de valores con órdenes de magnitud muy diferentes, entonces la minimización podría tender siempre a priorizar la minimización de los mismos objetivos en lugar de variar gradualmente con λ.

Cuando esto ocurre, es necesario normalizar las funciones objetivo, lo que cual resulta relativamente sencillo si se conocen los puntos ideal y nadir. Cada función objetivo f_k se puede normalizar como

$$f_k^{norm}(x) = \frac{f_k(x) - y_k^I}{y_k^N - y_k^I},$$

verificándose que $0 \leq f_k^{norm}(x) \leq 1$, para toda $k = 1, \ldots, p$ y $x \in \mathcal{X}_E$.

El cómputo del punto ideal es considerado un problema simple desde la perspectiva

de la optimización multiobjetivo, pues simplemente involucra resolver p problemas de optimización con un único objetivo. Por el contrario, determinar el punto nadir requiere optimizar sobre el conjunto eficiente, lo cual es un problema mucho más complejo. De hecho, no se conocen métodos eficientes para encontrarlo en problemas de optimización multiobjetivo generales. Por este motivo, en la práctica se recurren a heurísticas, tales como la estimación con tablas de *pay-off*; la cual se describe a continuación.

En primer lugar, para cada $k = 1, \ldots, p$, se resuelve el problema de optimización

$$\min_{x \in \mathcal{X}} f_k(x),$$

obteniendo una solución óptima denotada por x^k. Luego, para cada $i = 1, \ldots, p$, se evalúa la función objetivo f_i sobre todas las soluciones óptimas encontradas, es decir, $f_i(x^k)$ para todo $k = 1, \ldots, p$, y se selecciona el mayor de estos valores como la estimación de la i-ésima componente del punto nadir. Es decir, se define

$$\tilde{y}_i^N := \max_{k=1,\ldots,p} f_i(x^k)$$

como estimación del punto nadir. Es sabido que esta estimación puede sobrestimar o subestimar el valor real cuando hay más de dos objetivos presentes y cuando hay múltiples soluciones óptimas en los problemas $\min_{x \in \mathcal{X}} f_k(x)$. A pesar de esto, esta estimación es muy utilizada en la práctica.

La optimalidad de Pareto es la definición más difundidad de optimalidad en la optimización multiobjetivo, pero también es posible encontrar otras. Por ejemplo, la minimización puede adoptar una interpretación lexicográfica basada en un jerarquía de objetivos. Sin pérdida de generalidad, considerando el orden de prioridad $f_1 < \ldots < f_p$, se dice que una solución factible $\hat{x} \in \mathcal{X}$ es *lexicográficamente óptima* si no existe $x \in \mathcal{X}$ tal que $f(x) <_{lex} f(\hat{x})$ (donde, dados $y^1, y^2 \in \mathbb{R}^p$, $y^1 <_{lex} y^2$ si y solo si existe k^* en $\{1, \ldots, p\}$ tal que $y_k^1 = y_k^2$ para $k = 1, \ldots, k^* - 1$ e $y_{k^*}^1 < y_{k^*}^2$).

La jerarquía de objetivos en la optimización lexicográfica permite una resolución por optimización secuencial, minimizando un único objetivo f_k a la vez y usando el valor objetivo óptimo de f_i, con $i < k$, como restricción de la formulación, es

decir,

$$\begin{aligned} \text{mín} \quad & f_k(x) \\ \text{sujeto a} \quad & x \in \mathcal{X} \\ & f_i(x) = f_i(x^i) \quad i = 1, \ldots, k-1, \end{aligned}$$

donde x^i es la solución óptima del i-ésimo problema de optimización.