Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales

Algoritmos basados en Programación Lineal Entera para el Survivable Routing and Spectrum Assignment Problem

Tesis de Licenciatura en Ciencias de la Computación

Juan Pablo Lebon

Director: Dr. Javier Marenco
Codirectora: Dra. Flavia Bonomo
Buenos Aires, 2024

Abstract

Fiber optic networks use light, transported through a cable, as a means of communication between two *nodes* in a network. In response to the sustained growth of traffic demands in recent years, a new generation of fiber optic networks called *flexgrid elastic optical networks (EONs)* has been proposed, with the goal of making better use of the electromagnetic spectrum and in turn improving the capacity of such networks.

EONs divide the frequency spectrum of an optic fiber cable into narrow frequency *slots*, each with fixed bandwidth. Any consecutive sequence of slots can be used to form a simple channel, which in turn can be switched (or routed) in the network to establish a *lightpath*.

Given a set of demands and the topology of a network, the *routing and spectrum assignment problem (RSA)* consists of establishing lightpaths for a set of demands, which are expressed in terms of a source node, a target node, and a required amount of slots.

The *survivable RSA with path protection* problem is a variant of RSA, in which each demand is assigned two lightpaths: a *main* path, and a *backup* path, to provide tolerance for link failures in the network. This problem is $\mathcal{NP}$-hard, and has received much attention in recent years.

In this work, several Integer Programming models are proposed to tackle this problem, and their performance on real-world instances is studied. Aiming to improve the efficiency of such models, primal heuristics are proposed to accelerate the search of initial feasible solutions. Noting that the problem can be decomposed into a routing stage and an assignment stage, we propose alternative schemes to find solutions to these models based on Benders' decomposition.

Keywords: Integer Programming, Operations Research, Primal Heuristics, Benders Decomposition, Fiber Optic Networks, Routing and Spectrum Assignment.

Abstract

Las *redes de fibra óptica* utilizan la luz, transportada por un cable, como un medio de comunicación entre dos *nodos* de la red. En respuesta al crecimiento sostenido del tráfico en este tipo de redes, en los últimos años se ha propuesto una nueva generación de redes de fibra óptica, llamada *flexgrid elastic optical networks* (EONs) con el objetivo de mejorar la eficiencia en el uso del espectro electromagnético y aumentar así la capacidad de las redes.

En las EONs, el espectro de frecuencias de una fibra óptica se divide en *slots* de frecuencias relativamente pequeños, cada uno con un ancho de banda fijo. Se puede utilizar cualquier secuencia de slots consecutivos para formar un *canal*, que a su vez puede ser ruteado por la red para crear lo que se conoce como un *lightpath*.

Dada la estructura de una red y un conjunto de demandas, el *routing and spectrum assignment (RSA) problem* consiste en establecer los lightpaths para un conjunto de demandas de tráfico, cada una de las cuales está expresada en términos de un nodo de origen, un nodo de destino y una cantidad de slots. Dado que cada *lightpath* está determinado por una ruta y un canal, el RSA consiste en encontrar una ruta y asignar un intervalo de slots para cada demanda.

El *survivable RSA with path protection* es una variante de RSA, que corresponde a solicitar dos *lightpaths* para cada demanda: un *camino titular* y un *camino de backup*, y ambos deben respetar las restricciones de RSA. Este problema es $\mathcal{NP}$-hard y ha recibido atención por parte de la comunidad especializada en los últimos años.

En este trabajo se proponen distintos modelos de programación lineal entera para este problema, y se estudia su performance en la práctica sobre topologías reales. Se presentan además heurísticas para optimizar los modelos, buscando acelerar la obtención de soluciones factibles iniciales. Notando que el problema se puede descomponer en una fase de ruteo y una fase de asignación, se estudian esquemas de descomposición basados en la descomposición combinatoria de Benders para obtener soluciones a estos modelos mucho más rápidamente.

Palabras clave: Programación Lineal Entera, Investigación Operativa, Heurísticas Primales, Descomposición de Benders, Redes de Fibra Óptica, Ruteo y Asignación de Espectro.

Agradecimientos

A mis directores, Javi y Flavia, por ser una enorme fuente de inspiración y conocimiento durante todo este proceso. Les agradezco por todo lo que me compartieron, lo técnico y lo humano.

A papá, mamá, y a Juli, quienes me dieron absolutamente todo lo que podría pedir en la vida y siempre confiaron en mí.

A Mica, el amor de mi vida, la compañía que toda la vida busqué. No puedo resumir en un párrafo lo que significás para mí; más bien llevaría años, así que busco expresarlo de a poco todos los días.

A los Mushus, por el apoyo y la amistad durante todos estos años. A Santi, mi compañero de TPs de principio a fin. A Alan y Lucas, que completan el cuarteto del final de la carrera: llegar a la facultad y estar con ustedes todos los días fue una fuente infinita de alegría. A Gonza, por tu constante calidez y por las sesiones maratónicas de estudio en cualquier pizarrón libre que encontráramos. A Canta, por compartir tantos de mis intereses, por entenderme. A Rosen, por ser la persona más buena del mundo. A Lu, Nico, Dete, Cielo, Juani, Max, Alex, y Eitu, que completan un grupo que no sería lo mismo sin cada uno de ustedes.

A Max y Emma, mis dos ratitas, por interminable afecto y su compañía en mi cuarto durante tantas sesiones de estudio.

A la hermosa Facultad de Ciencias Exactas y Naturales, por hacerme sentir todos los días que el cambio de carrera valió la pena.

Índice general

Parte I

Capítulo 1

Introducción

1.1. Orígenes

La optimización combinatoria es un área en la intersección de la computación y la matemática que busca encontrar, a partir de un conjunto discreto de soluciones a un problema, la mejor de estas soluciones según un objetivo a cumplir. Dada la multitud de problemas del mundo real que pueden ser expresados como problemas de optimización combinatoria – planificar un ruteo de camiones de entrega para minimizar el uso de recursos, determinar cuánto producir de una serie de productos para maximizar ganancias, entre muchos otros – mucha atención y tiempo se ha dedicado en las últimas décadas a encontrar herramientas para resolverlos eficientemente. En este trabajo nos concentramos en dos herramientas que resultaron ser un punto de inflexión en la historia de la optimización combinatoria: la *programación lineal*, y la *programación lineal entera*.

1.1.1. Programación Lineal

En 1947 George B. Dantzig, matemático famoso y en ese momento un consultor para la Fuerza Aérea de los Estados Unidos, trabajó sobre el problema de diseñar un programa de suministro logístico, entrenamiento y despliegue de tropas. En particular, se buscaba una manera de agilizar el proceso de obtención del mismo, el mejor caso posible siendo un algoritmo que pudiera ser ejecutado por una computadora. Inspirado en el trabajo del economista Wassily Leontief de 1932 [1], que modelaba la economía estadounidense a través de una representación matricial de las interdependencias de sus distintos sectores, Dantzig formuló el problema en cuestión como un sistema que contenía:

1. un conjunto de desigualdades lineales, que representan las interdependencias del objeto de estudio y las restricciones que imponían,

2. una función objetivo lineal a maximizar, usada para elegir (con algún criterio basado en experiencia de mundo real) la mejor solución entre todas las posibles.

Dantzig describió este modelo en un paper llamado "*Programming in a Linear Structure*". Pocas horas antes de presentarlo en una conferencia en RAND, y a sugerencia de un colega con el que en ese momento estaba paseando, le dio al método un nuevo nombre: "*Linear Programming*".

El trabajo que le dedicó en los siguientes meses para resolver estos modelos de manera automatizada resultó en el diseño del método Simplex. Este método, creado a finales de la

década del '40, sigue siendo el estándar hoy en día para resolver modelos de programación lineal; si bien tiene complejidad temporal exponencial en peor caso, funciona muy bien en la práctica, en muchas instancias incluso mejor que los algoritmos polinomiales que a lo largo de las décadas fueron hallados para el problema.

1.1.2. Programación Lineal Entera

Diez años después del trabajo inicial de Dantzig, el matemático Ralph E. Gomory desarrolló una variante del algoritmo Simplex para hallar soluciones enteras a modelos de programación lineal. En ese momento, por su rol de consultor de las Fuerzas Navales Estadounidenses, Gomory había asistido a una charla en Washington sobre un problema de programación lineal que modelaba un *Task Force* de la Marina. Durante la charla, uno de los oradores comentó que sería útil recibir soluciones enteras al mismo, ya que el problema hablaba de cantidades de soldados y portaaviones a despachar.

Inspirado por el comentario, y con la intuición de que si uno estuviera buscando resolver el problema a mano, podría resolver la versión fraccionaria del problema y usar la solución hallada como cota a la solución entera, Gomory diseñó en el 1958 un algoritmo para obtener soluciones enteras [2], utilizando lo que llamó "*cortes fraccionarios*". Este resultado rápidamente generó mucho interés en la comunidad, por su gran potencial de aplicación en la optimización de procesos de muchas industrias más allá del ámbito militar.

Formulaciones que expresan problemas de optimización combinatoria en términos de una función objetivo lineal, junto a un sistema de restricciones lineales a cumplir:

$$\begin{aligned} \max \quad & cx \\ \text{s.a.} \quad & Ax \leq b, \\ & x \geq 0 \end{aligned}$$

se denominan modelos de programación lineal (PL). La variante de estos modelos en la que las variables están restringidas a valores enteros se denominan modelos de programación lineal entera (PLE). Si bien estos últimos permiten fácilmente modelar problemas en los que uno quiere obtener soluciones enteras, la mayor ventaja que ofrecen está en su poder expresivo: al poder restringir variables a valores binarios, se puede utilizar la programación lineal entera para modelar muy naturalmente problemas de decisión.

Es importante notar que estas ventajas incurren costo computacional; si bien se conocen algoritmos polinomiales para resolver PL, PLE pertenece a la clase $\mathcal{NP}$-Hard [3], y luego no se conocen algoritmos polinomiales para resolverlo. No obstante, distintas herramientas a lo largo de las décadas han demostrado su utilidad para resolverlos eficientemente en la práctica. En la década del '50, Gomory publicó los primeros algoritmos para obtener de manera automatizada planos de corte para un modelo dado, que reducen el espacio de soluciones de un problema. Estos incluyen los llamados *cortes fraccionarios* [2], y *cortes mixtos* [4]. Trabajos posteriores incluyen los *cortes disyuntivos* de Balas [5] en el 1979, las *desigualdades de cubrimiento* o *cover inequalities* de Balas [6] y Wolsey [7] en el 1975 y subsecuentes optimizaciones por Crowder et al. [8] y Van Roy et al. [9] en la década del '80, y los cortes *Lift-and-Project* [10] de Balas et al. en 1993. La demostrada utilidad de los planos de corte ha llevado, además, a que sea estándar la formulación de desigualdades válidas ad-hoc, donde en vez de obtener planos de corte genéricos con algoritmos de propósito general, se reflexiona sobre propiedades particulares del problema en cuestión y se busca introducir dichas propiedades al modelo en la forma de restricciones adicionales.

Avances teóricos como estos, junto al incremento exponencial en poder de cómputo y al desarrollo de *solvers* de alta calidad (tanto propietarios como de código abierto), han posicionado a la programación lineal entera como la herramienta por excelencia para problemas de optimización combinatoria, y hoy en día son ubicuas al momento de resolver estos problemas en distintas industrias, además de ser de gran interés académico.

1.1.3. Redes de fibra óptica

Los cables de fibra óptica, construidos con vidrio o plástico, utilizan la emisión de luz como medio de comunicación entre nodos emisores y receptores. Este tipo de cable presenta varias ventajas en comparación a tecnologías anteriores como los cables de cobre y los cables coaxiales. Dado que las señales emitidas por este medio viajan a velocidades cercanas a un 70 % de la velocidad de la luz, estos cables admiten velocidades de transmisión mucho mayores que otros medios. Su construcción física permite además enviar señales con atenuación muy baja y poca interferencia externa, lo cual lo hace un medio ideal para establecer conexiones a larga distancia. En la figura 1.1 se pueden ver dos ejemplos de topologías reales, y el gran territorio que pueden llegar a abarcar.

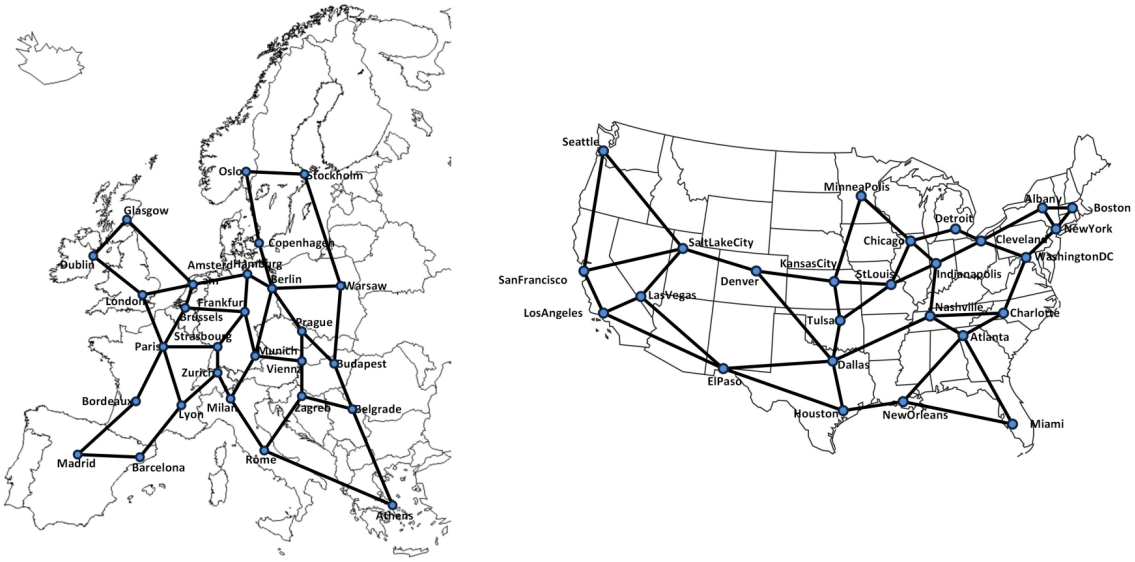

Figura 1.1: *Ejemplos de topologías de redes de fibra óptica: Euro28 (izquierda) y US26 (derecha) [11].*

La velocidad y robustez que ofrece esta tecnología la ha posicionado como una herramienta crucial en las telecomunicaciones modernas. A lo largo de los años, además, se han desarrollado avances para explotar al máximo las capacidades del medio. Para transportar múltiples señales a la vez a lo largo de una única fibra se utiliza la *multiplexación por división de longitud de ondas* (o WDM, por sus siglas en inglés). Aquí, señales entrantes de distintas frecuencias son multiplexadas para viajar por un mismo cable, y luego son desmultiplexadas al llegar a un nodo receptor. Esto permite multiplicar la capacidad de transmisión de los cables, y así es que se volvió de uso ubicuo en este tipo de redes.

Una optimización adicional que recibió adopción masiva es la tecnología *fixed grid*, que divide el espectro electromagnético en secciones – denominadas canales – típicamente de 50 o 100 GHz. Esta división simplifica el diseño de las redes y facilita la integración con

protocolos estandarizados de transferencia múltiple de señales, como *Synchronous Digital Hierarchy*.

Las últimas tres décadas contaron con la tecnología fixed grid para optimizar el uso de los recursos en las redes de fibra óptica. En los últimos años, sin embargo, este enfoque se ha mostrado cada vez más incapaz de manejar las crecientes demandas del día a día. La llegada y popularización global de servicios de *streaming* e Internet Protocol Television, de almacenamiento en la nube, y de dispositivos móviles en constante comunicación, entre otros, no solo han incrementado los requerimientos de capacidad de las redes de telecomunicaciones, sino que también han aumentado la diversidad en el tipo de demanda que estas redes deben enfrentar. Esto se debe a que, por ejemplo, no requiere igual ancho de banda[1] el envío de un mensaje desde un teléfono celular que la descarga de un video de alta resolución. La creciente diversidad en las demandas resalta las limitaciones de la tecnología fixed grid; al reservar iguales porciones del espectro para cualquier tipo de demanda, sin importar la magnitud de sus requerimientos, se termina realizando un uso subóptimo de los recursos.

Las redes ópticas elásticas (o EONs, por sus siglas en inglés) apuntan a enmendar esto. Utilizando tecnología de *flexible grid*, las EONs pueden definir canales de longitud variable que puedan ajustarse dinámicamente a las demandas entrantes, evitando así asignar recursos excesivos a situaciones que no los requieran. Esta tecnología divide el espectro en secciones mucho más chicas, típicamente de 12.5 GHz, denominadas *slots*. Para atender una demanda particular, se asigna una cantidad apropiada de slots consecutivos para definir el canal a través del cual dicha demanda será enviada. La utilidad de este ahorro de recursos se puede observar en la figura 1.2, que compara el uso de espectro al utilizar fixed grid como tecnología subyacente y utilizando flexible grid. Se puede ver allí que aumenta no solamente la cantidad de demandas que pueden ser transmitidas en simultáneo, si no también el máximo volumen de demanda que puede ser transmitido.

1.2. El problema de Ruteo y Asignación de Espectro

Dada la estructura de una red y un conjunto de demandas, el *routing and spectrum assignment (RSA) problem* consiste en establecer los lightpaths para un conjunto de demandas de tráfico, cada una de las cuales está expresada en términos de un nodo de origen, un nodo de destino y una cantidad de slots. Dado que cada *lightpath* está determinado por una ruta y un canal, el RSA consiste en encontrar una ruta y asignar un intervalo de slots para cada demanda. Para operar adecuadamente la red, se deben tener en cuenta las siguientes restricciones:

1. *continuidad*: los slots utilizados deben ser los mismos en todos los enlaces de cada ruta;

2. *contigüidad*: los slots asignados a una demanda deben ser contiguos;

3. *no-solapamiento*: en cada enlace, cada slot debe ser asignado a lo sumo a una demanda.

[1] El ancho de banda es una medida de la capacidad de transmisión de información que tiene un medio. Cuanto mayor es el ancho de banda, más capacidad de transmisión tiene. El ancho de banda es una medida de volumen de datos, y no debe confundirse con la *velocidad* de transmisión del medio.

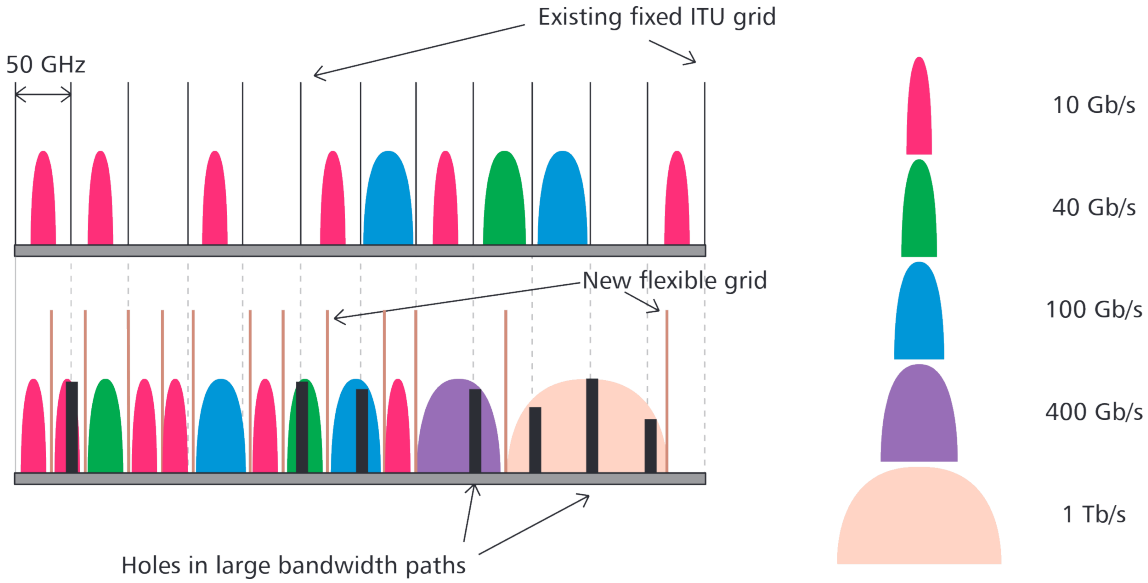

Figura 1.2: *Asignación de recursos para un conjunto de demandas de distintos tamaños, en redes fixed grid (arriba) y flexible grid (abajo) [12]*

La figura 1.3 muestra una solución a una instancia del RSA de ejemplo en una topología reducida. Notar que cada demanda envía el volumen requerido desde su nodo origen a su nodo destino, y que todo par de demandas cuyos caminos se cruzan reciben conjuntos de slots disjuntos.

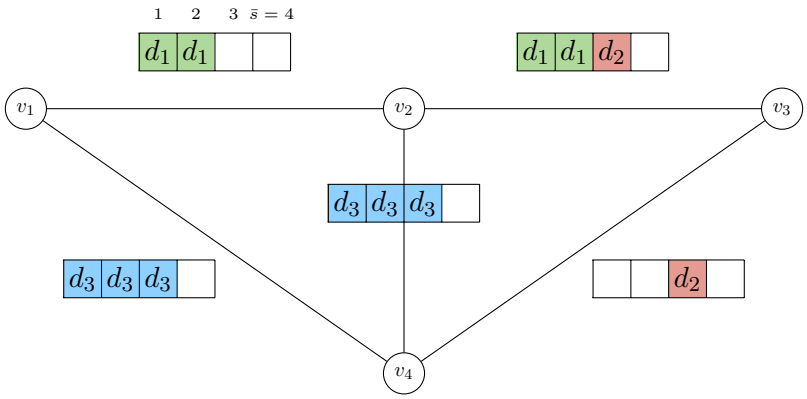

Figura 1.3: *Solución al problema para las demandas* $d_1 = (v_1, v_3, 2), d_2 = (v_2, v_4, 1)$, *y* $d_3 = (v_1, v_2, 3)$.

El problema de RSA es notablemente difícil desde un punto de vista teórico: no solo pertenece a la clase $\mathcal{NP}$*-Hard* [13] [14], sino que también es $\mathcal{NP}$*-Hard* una versión reducida del problema en la que todos los caminos de las demandas se conocen de antemano.

El *survivable RSA with path protection* (S-RSA) es una variante de RSA, que corresponde a solicitar dos *lightpaths* para cada demanda: un *camino titular* y un *camino de backup* (con una fracción de los slots demandados originalmente), que respeten las restricciones de RSA y que usen el mismo conjunto de slots. Este problema, que generaliza RSA, también es $\mathcal{NP}$*-Hard* y ha recibido atención por parte de la comunidad especializada

en los últimos años. En este trabajo estudiaremos dos versiones del S-RSA, que buscan implementar la protección de caminos de distintas maneras.

La primera estrategia, denominada *Dedicated Path Protection* (DPP), asigna los recursos de respaldo de manera exclusiva a cada demanda, de manera que si una demanda usa un cierto camino c como camino de respaldo, ninguna otra demanda puede utilizar arcos de c en sus caminos de respaldo. La figura 1.4 ilustra las reglas que este tipo de protección impone sobre los caminos de cada demanda.

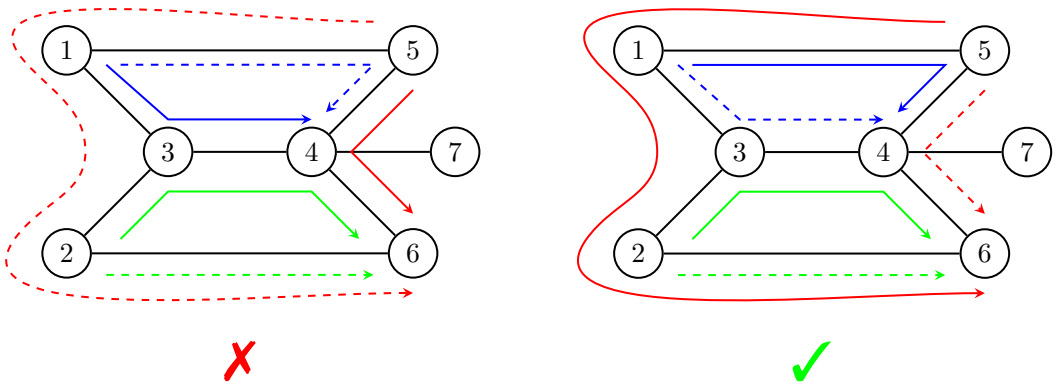

Figura 1.4: *Figura ejemplificando la correcta e incorrecta implementación del ruteo de tres demandas (roja, azul, y verde) con protección vía caminos dedicados. Las líneas lisas representan caminos titulares y las líneas punteadas caminos de respaldo. La figura izquierda es un intento erróneo de implementar este tipo de protección ya que las demandas comparten arcos en sus caminos de respaldo. La figura derecha, por otro lado, logra rutear las demandas de manera válida.*

Esta exclusividad trae beneficios en redes reales que son altamente deseables, como por ejemplo la capacidad de alternar rápidamente entre uno de los caminos asignados y el otro, de ser necesario. Además, al no incurrir congestión en los recursos de respaldo, es común tener configurados en la red ambos caminos a la vez, para permitir la alternación instantánea entre el camino titular y de respaldo de una demanda. La principal desventaja de este paradigma es la alta cantidad de recursos que pide; al requerir que los caminos de respaldo de cada demanda sean disjuntos entre sí, la cantidad de demandas que pueden ser ruteadas con este tipo de protección es baja. Cuando el objetivo principal de una red es permitir el flujo de cantidades de datos masivas, este esquema puede no ser ideal.

El segundo tipo de protección de caminos que consideraremos en este trabajo se denomina Shared Backup Path Protection (SBPP), en donde dos demandas tienen permitido compartir recursos en sus caminos de respaldo siempre y cuando sus caminos titulares sean disjuntos. Estas reglas se ilustran en la figura 1.5. Por un lado, este esquema tiene claras ventajas: el uso más eficiente de recursos permite transmitir de manera protegida un volumen de datos mucho mayor a través de las redes y además ofrece reducciones de costos significativas. Las desventajas de este esquema en comparación a DPP incluyen la tendencia a generar congestión en las redes en caso de múltiples fallas simulatáneas, y el hecho de que la activación de un camino de respaldo ante una falla es más lento que en DPP, por requerir una reconfiguración de los *switches* de la red. Esto puede ser un problema grave en casos de uso en donde la conectividad constante es crucial.

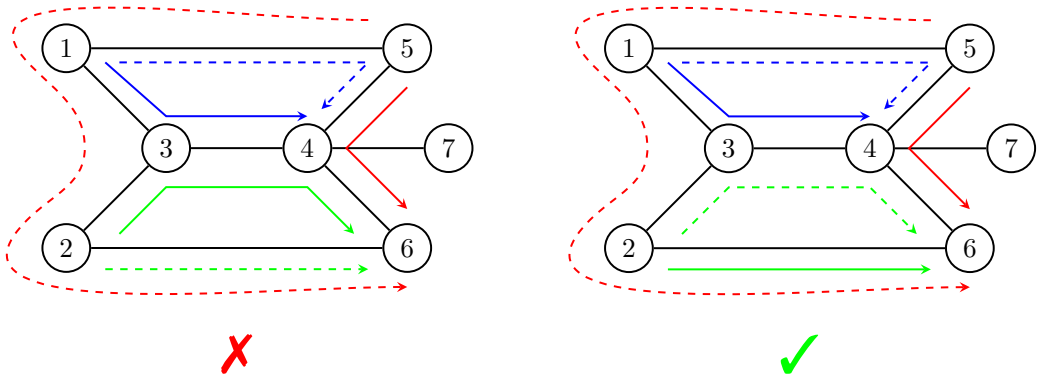

Figura 1.5: *Figura ejemplificando la correcta e incorrecta implementación del ruteo de tres demandas (roja, azul, y verde) con protección vía caminos de respaldo compartidos. Las lineas lisas representan caminos titulares y las lineas punteadas caminos de respaldo. La figura izquierda es un intento erróneo de implementar este tipo de protección ya que las demandas verde y roja comparten arcos en sus caminos de respaldo cuando sus caminos titulares no son disjuntos. La figura derecha, por otro lado, logra rutear las demandas de manera válida.*

1.3. Estructura del trabajo

En este capítulo introductorio discutimos la emergencia de las redes de fibra óptica, destacando la implementación vía tecnología *flexgrid*, y comentamos acerca del problema de Ruteo y Asignación de Espectro (RSA) que surge en esta última. Presentamos aquí adicionalmente la variante de RSA a la cual este trabajo está dedicada, el Survivable RSA, (S-RSA) que busca dotar a las redes con protección de caminos para que sean tolerantes a fallas.

El resto de este trabajo está estructurado de la siguiente manera. En el capítulo 2 diseñamos modelos de programación lineal entera para S-RSA en su versión con DPP y con SBPP. Estudiamos el desempeño computacional de los mismos y los comparamos contra el modelo base (que resuelve RSA), para obtener una noción de cuánto la protección de caminos empeora los tiempos de resolución. En el capítulo 3 buscamos acelerar la resolución de estos modelos vía heurísticas primales, cuyo objetivo es rápidamente obtener soluciones iniciales factibles para el problema, que el modelo completo pueda luego usar como *warm-start*. En dicho capítulo estudiamos la calidad de las soluciones producidas por las heurísticas, y el efecto que tienen sobre el tiempo completo de resolución del problema.

En la parte 2 de este trabajo, nos alejaremos del diseño de modelos para el S-RSA, buscando en cambio maneras alternativas de resolver los modelos previamente diseñados. Aquí consideraremos tanto RSA como S-RSA, buscando acelerar la resolución de DR-AOV y de los modelos que proponemos. El primer capítulo de esta parte, el capítulo 4 del trabajo, se concierne por el RSA, y el capítulo siguiente con el S-RSA.

Finalmente, en el capítulo 6 reflexionamos sobre los resultados obtenidos a lo largo del trabajo y proponemos distintas avenidas de trabajo futuro. Adicionalmente, en el apéndice detallamos algunos resultados computacionales secundarios que no incluimos en la parte principal del trabajo, pero que no obstante respaldan ciertas decisiones tomadas durante el mismo.

Capítulo 2

Modelos para el survivable RSA

En este capítulo diseñaremos modelos para el S-RSA bajo dos estrategias distintas de protección de caminos: protección dedicada, en donde los caminos de respaldo de cada demanda deben ser disjuntos de los caminos de respaldo de las otras demandas, y protección compartida, en la que dos demandas pueden compartir recursos en sus caminos de respaldo si sus caminos titulares son disjuntos. En ambos casos, el objetivo es asegurar la tolerancia a fallas en un enlace cualquiera de la red.

2.1. Modelos de la literatura

Los primeros modelos de programación lineal entera para atacar el RSA fueron publicados a finales de 2010 y principios de 2011. Estos incluyen una heurística basada en PLE por Christodoulopoulos et al. [15] y un modelo de Klinkowski et al. [16] que contenía variables para asociar demandas a caminos o a lightpaths en la red. Este segundo trabajo marcó una tendencia en los primeros modelos para el RSA; por un par de años, los modelos propuestos para el problema tenían variables para cada posible camino entre todo par origen-destino [17] [18], y luego estos crecían exponencialmente con el tamaño de la red. Para controlar el tamaño de los modelos, la estrategia estándar durante este tiempo fue precomputar un conjunto de posibles caminos a tomar para cada demanda [18] [19]. La situación se prestaba a un balance poco ideal: si se consideraban todos los posibles caminos para cada demanda el modelo resultante era impráctico de resolver, y si ciertos caminos se prohibían la solución bien podría no ser óptima.

Un cambio de paradigma llegó unos años después en la forma de formulaciones *Edge-Node* que definen caminos para las demandas de manera más granular, expresando las asociaciones en términos de los arcos individuales y no de los caminos enteros. Gracias a que las variables en estos modelos crecen de manera polinomial en el tamaño de la instancia, esta formulación se ha vuelto la más común en modelos modernos. Avances notables en esta corriente incluyen, en orden cronológico, el trabajo de Cai et al. [20], Dao Thanh. [21], Muhammad et al. [22], Bertero et al. [23], y Hadhbi et al. [24].

Los modelos que propondremos están fuertemente basados en el trabajo de Bertero, Bianchetti, y Marenco [23], y en particular en el modelo *Demand-Range Alternate Order Variable (DR-AOV)*. Este, como todo modelo de la clase *Demand-Range*, expresa la asignación de slots en términos del rango que estos cubren, teniendo variables l_d y r_d para el primer y último slot usado, respectivamente. Este fue elegido como punto de partida por presentar los mejores resultados computacionales en dicho trabajo, que comparó su

desempeño no solo contra el resto de los modelos propuestos por los autores sino también contra una variedad de modelos de la literatura.

2.2. Notación

A continuación describimos las variables y constantes que se usarán en los modelos. Recordemos que estos caen en la categoría "*Demand-Range*".

Constantes:

$\overline{s} \in \mathbb{N}$	la cantidad de slots de cada arco
$S = \{1, \ldots, \overline{s}\}$	el conjunto de slots
$v(d) \in \mathbb{N}$	el volumen de la demanda d
$s(d) \in V$	la fuente de la demanda d
$t(d) \in V$	el destino de la demanda d

Variables

$y_{de} \in \{0, 1\}$	igual a 1 si la demanda d usa el arco e
$l_d \in \mathbb{N}$	el índice del primer slot usado por d
$r_d \in \mathbb{N}$	el índice del último slot usado por d
$n^{mm}_{dd'} \in \{0, 1\}$	igual a 1 si d y d' usan un mismo arco y $r_d < l_{d'}$
$ym_{de} \in \{0, 1\}$	igual a 1 si la demanda d usa el arco e en su camino titular
$yb_{de} \in \{0, 1\}$	igual a 1 si la demanda d usa el arco e en su camino de respaldo
$lm_d \in \mathbb{N}$	el índice del primer slot usado por d en su camino titular
$lb_d \in \mathbb{N}$	el índice del primer slot usado por d en su camino de respaldo
$rm_d \in \mathbb{N}$	el índice del último slot usado por d en su camino titular
$rb_d \in \mathbb{N}$	el índice del último slot usado por d en su camino de respaldo
$n^{mm}_{dd'} \in \{0, 1\}$	igual a 1 si $rm_d < lm_{d'}$
$n^{mb}_{dd'} \in \{0, 1\}$	igual a 1 si $rm_d < lb_{d'}$
$n^{bm}_{dd'} \in \{0, 1\}$	igual a 1 si $rb_d < lm_{d'}$
$n^{bb}_{dd'} \in \{0, 1\}$	igual a 1 si $rb_d < lb_{d'}$
$p_{dd'} \in \{0, 1\}$	igual a 1 los caminos titulares de d y d' son disjuntos

2.3. Modelo base

2.3.1. DR-AOV

El modelo propuesto por los autores, que actuará de punto de partida, busca por un lado rutear las demandas y por otro lado configurar los slots por los que viajará cada una dentro de su camino asignado. El ruteo se realiza imponiendo restricciones de flujo, y se busca minimizar sus longitudes con la función objetivo. Como mencionamos anteriormente, la configuración de slots se implementa con variables l_d y r_d, que expresan respectivamente el primer y último slot que utilizará la demanda d.

$$\min \quad \sum_{d \in D} \sum_{e \in E} y_{de}$$

$$\text{s.a.} \sum_{e \in \delta^+(v)} y_{de} - \sum_{e \in \delta^-(v)} y_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{en caso contrario} \end{cases}, \qquad \forall d \in D, v \in V \tag{2.1}$$

$$n_{dd'} + n_{d'd} \geq y_{de} + y_{d'e} - 1, \qquad \forall d, d' \in D, e \in E \tag{2.2}$$

$$l_d + v(d) \leq l_{d'} + \overline{s}(1 - n_{dd'}), \qquad \forall d, d' \in D \tag{2.3}$$

$$r_d = l_d + v(d) - 1, \qquad \forall d \in D \tag{2.4}$$

$$1 \leq l_d \leq \overline{s} - v(d) + 1, \qquad \forall d \in D \tag{2.5}$$

$$y_{de} \in \{0, 1\}, \qquad \forall d \in D, e \in E \tag{2.6}$$

$$l_d, r_d \in \mathbb{Z}, \qquad \forall d \in D \tag{2.7}$$

La función objetivo busca minimizar el uso de arcos de la red. En (2.1) se plantean restricciones de flujo que aseguran que haya un camino desde la fuente de cada demanda hasta su destino. Esta restricción realiza el enrutamiento de las demandas. La restricción (2.2) pide que si dos demandas comparten un arco, los slots de alguna estén a la izquierda que los slots de la otra. Las restricciones (2.3) y (2.4) definen, en términos de los rangos dados por l y r, qué significa que un conjunto de slots esté a la izquierda de otro. Estas restricciones se encargan de armar una configuración de slots donde no haya solapamiento entre demandas. La restricción (2.5) fuerza que los slots asignados a cada demanda no caigan fuera de rango.

2.4. Modelos con protección de caminos

A continuación presentamos los modelos que robustecen a DR-AOV con protección de caminos dedicada y compartida. A un alto nivel, la estrategia para construirlos es la misma en ambos casos: se definen restricciones para los lightpaths titulares y de respaldo para que estos sean válidos individualmente, y luego se agregan restricciones adicionales que impongan la separación apropiada entre dichos lightpaths según el tipo de protección buscada. En el caso de protección dedicada, agregaremos restricciones que prohíban el uso compartido de recursos de respaldo entre dos demandas cualesquiera. En el caso de protección compartida, agregaremos restricciones que permitan que dos demandas compartan recursos si sus caminos titulares son disjuntos.

2.4.1. DPP

El siguiente modelo adapta DR-AOV para que cuente con protección de caminos dedicada. Como mencionamos anteriormente, buscaremos por un lado definir restricciones análogas para los caminos titulares y de respaldo de manera que individualmente se formen lightpaths válidos, y luego agregaremos restricciones adicionales para que no haya conflictos entre ellos. En este caso, las restricciones adicionales prohibirán que las demandas compartan recursos en sus caminos de respaldo.

$$\min \quad \sum_{d\in D}\sum_{e\in E} (ym_{de} + yb_{de})$$

$$\text{s.a.} \sum_{e\in\delta^+(v)} ym_{de} - \sum_{e\in\delta^-(v)} ym_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{otherwise} \end{cases}, \quad \forall d \in D, v \in V \tag{2.8}$$

$$\sum_{e\in\delta^+(v)} yb_{de} - \sum_{e\in\delta^-(v)} yb_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{otherwise} \end{cases}, \quad \forall d \in D, v \in V \tag{2.9}$$

$$ym_{de} + yb_{de} \leq 1, \quad \forall d \in D, e \in E \tag{2.10}$$

$$\sum_{d\in D} yb_{de} \leq 1, \quad \forall e \in E \tag{2.11}$$

$$n^{mm}_{dd'} + n^{mm}_{d'd} \geq ym_{de} + ym_{d'e} - 1, \quad \forall d, d' \in D, e \in E \tag{2.12}$$

$$n^{mb}_{dd'} + n^{bm}_{d'd} \geq ym_{de} + yb_{d'e} - 1, \quad \forall d, d' \in D, e \in E \tag{2.13}$$

$$lm_d + v(d) \leq lm_{d'} + \overline{s}(1 - n^{mm}_{dd'}), \quad \forall d, d' \in D \tag{2.14}$$

$$lm_d + v(d) \leq lb_{d'} + \overline{s}(1 - n^{mb}_{dd'}), \quad \forall d, d' \in D \tag{2.15}$$

$$lb_d + v(d) \leq lm_{d'} + \overline{s}(1 - n^{bm}_{dd'}), \quad \forall d, d' \in D \tag{2.16}$$

$$rm_d = lm_d + v(d) - 1, \quad \forall d \in D \tag{2.17}$$

$$rb_d = lb_d + v(d) - 1, \quad \forall d \in D \tag{2.18}$$

$$lm_d \geq 1, \quad \forall d \in D \tag{2.19}$$

$$lb_d \geq 1, \quad \forall d \in D \tag{2.20}$$

$$lm_d \leq \overline{s} - v(d) + 1, \quad \forall d \in D \tag{2.21}$$

$$lb_d \leq \overline{s} - v(d) + 1, \quad \forall d \in D \tag{2.22}$$

Las restricciones (2.8) y (2.9) plantean restricciones de flujo para los caminos titulares y de respaldo de cada demanda. Estas, junto a la función objetivo que minimiza el uso de arcos, garantizan que cada demanda será asignado exactamente un camino conexo desde su origen a su destino, y que este será lo más corto posible. La restricción (2.10) garantiza que los caminos titulares y backup de cada demanda sean disjuntos. La restricción (2.11) impide que dos demandas compartan recursos en sus caminos de respaldo. Es decir, impiden que usen un mismo arco en sus caminos de respaldo. Las restricciones (2.12) y (2.13) piden que si dos demandas comparten un arco (ya sea en sus caminos titulares, o en el camino titular de una y en el backup de la otra), entonces los slots asignados a una demanda estén a la izquierda de los slots asignados a la otra. Las restricciones (2.14) a (2.16) implementan la separación de slots planteada en las restricciones del ítem anterior.

2.4.2. SBPP

El modelo que implementa la protección de caminos compartida es similar al anterior; al igual que para DPP, implementaremos restricciones para definir lightpaths titulares y de respaldo válidos y además nos aseguraremos que no haya conflictos entre ellos. En este caso, lo que prohibimos es el uso compartido de recursos de respaldo cuando las demandas asociadas comparten recursos en sus respectivos caminos titulares. Así, empezamos a

considerar adicionalmente variables que expresan cuándo dos demandas pueden compartir recursos en sus caminos de respaldo. Buscaremos identificar con una nueva variable p cuáles demandas tienen caminos titulares disjuntos, y serán estos los que tendrán permitido compartir recursos de respaldo.

$$\min \quad \sum_{d\in D}\sum_{e\in E} (ym_{de} + yb_{de})$$

$$\text{s.a.} \sum_{e\in\delta^+(v)} ym_{de} - \sum_{e\in\delta^-(v)} ym_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{otherwise} \end{cases}, \quad \forall d \in D, v \in V \tag{2.23}$$

$$\sum_{e\in\delta^+(v)} yb_{de} - \sum_{e\in\delta^-(v)} yb_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{otherwise} \end{cases}, \quad \forall d \in D, v \in V \tag{2.24}$$

$$ym_{de} + yb_{de} \leq 1, \quad \forall d \in D, e \in E \tag{2.25}$$

$$n^{mm}_{dd'} + n^{mm}_{d'd} \geq ym_{de} + ym_{d'e} - 1, \quad \forall d, d' \in D, e \in E \tag{2.26}$$

$$n^{mb}_{dd'} + n^{bm}_{d'd} \geq ym_{de} + yb_{d'e} - 1, \quad \forall d, d' \in D, e \in E \tag{2.27}$$

$$n^{bb}_{dd'} + n^{bb}_{d'd} \geq yb_{de} + yb_{d'e} - 1, \quad \forall d, d' \in D, e \in E \tag{2.28}$$

$$lm_d + v(d) \leq lm_{d'} + \overline{s}(1 - n^{mm}_{dd'}), \quad \forall d, d' \in D \tag{2.29}$$

$$lm_d + v(d) \leq lb_{d'} + \overline{s}(1 - n^{mb}_{dd'}), \quad \forall d, d' \in D \tag{2.30}$$

$$lb_d + v(d) \leq lm_{d'} + \overline{s}(1 - n^{bm}_{dd'}), \quad \forall d, d' \in D \tag{2.31}$$

$$lb_d + v(d) \leq lb_{d'} + \overline{s}(1 - n^{bb}_{dd'}), \quad \forall d, d' \in D \tag{2.32}$$

$$p_{dd'} \geq yb_{de} + yb_{d'e} - 1, \quad \forall d, d' \in D, e \in E \tag{2.33}$$

$$2 - \sum_{e\in\delta^-(v)} ym_{de} - \sum_{e\in\delta^-(v)} ym_{d'e} \geq p_{dd'}, \quad \forall d, d' \in D, v \in V \tag{2.34}$$

$$rm_d = lm_d + v(d) - 1, \quad \forall d \in D \tag{2.35}$$

$$rb_d = lb_d + v(d) - 1, \quad \forall d \in D \tag{2.36}$$

$$lm_d \geq 1, \quad \forall d \in D \tag{2.37}$$

$$lb_d \geq 1, \quad \forall d \in D \tag{2.38}$$

$$lm_d \leq \overline{s} - v(d) + 1, \quad \forall d \in D \tag{2.39}$$

$$lb_d \leq \overline{s} - v(d) + 1, \quad \forall d \in D \tag{2.40}$$

En este modelo se elimina la restricción (2.11) de DPP, que impedía que dos demandas compartan recursos en sus caminos de respaldo. Se agrega la restricción (2.28) que pide que si los caminos de respaldo de dos demandas se cruzan, los slots de una estén a la izquierda de los de la otra. Se agrega la restricción (2.32), que busca obtener la separación de slots para caminos de respaldo de dos demandas cuando uno de estos conjuntos pretende estar a la izquierda del otro. Se agregan las restricciones (2.33) y (2.34), que prohíben que dos demandas compartan recursos en sus caminos de respaldo si sus caminos titulares no son disjuntos. Si en la restricción (2.33) se intenta definir yb_{de} y $yb_{d'e}$ ambas en 1 – así indicando que las demandas d y d' compartirían recursos – se buscará asignarle valor 1 a $p_{dd'}$, y es la restricción (2.34) quien encargará de determinar si eso es posible o no

dependiendo de si las demandas comparten recursos en sus caminos titulares. Cuando este uso compartido ocurre, existe algún nodo v al cual d y d' ambas entran en sus caminos titulares[1], y en consecuencia el lado izquierdo de la desigualdad valdrá cero y forzará a que $p_{dd'}$ también sea nulo.

En resumen, este modelo refina DPP definiendo nuevas reglas que permiten el uso compartido recursos en caminos de respaldo. En vez de prohibirlo en todo caso, ahora se permite siempre y cuando los caminos titulares sean disjuntos. En este caso, se definen para estos caminos las reglas usuales de configuraciones de slots.

2.5. Instancias de prueba

En este trabajo utilizamos una variedad de topologías de la literatura, que representan redes reales de distintos tamaños. La menor de ellas tiene 6 nodos y 9 arcos, y la mayor cuenta con 43 nodos y 176 arcos.

Para evaluar el rendimiento de los modelos propuestos trabajamos con una amplia suite de tests, con muchas configuraciones de cantidad de demandas y slots, y de volúmenes de demanda. Los tamaños de D y $\overline{s}$ se dividen en tres categorías:

- Tamaño chico: $|D| \in \{10,\ 20,\ 40\}$, $\overline{s} \in \{2|D|,\ 2|D|+10,\ 2|D|+20,\ 2|D|+30\}$
- Tamaño mediano: $|D| \in \{30,\ 50,\ 80\}$, $\overline{s} = 32$
- Tamaño grande: $|D| \in \{100,\ 150,\ 180\}$, $\overline{s} = 320$.

Las instancias de tamaño mediano y grande están basadas en propiedades de redes reales, donde los slots tienen ancho de banda de 12.5GHz y los arcos tienen, dependiendo de la red, ancho de banda de 400GHz o 4000GHz. Las instancias de tamaño chico buscan conseguir una visión más completa del rendimiento de los modelos, permitiendo analizarlos en instancias donde la esperanza de factibilidad es grande aún con volúmenes de demandas más altos.

Consideraremos también distintos volúmenes de demandas. En primer lugar tendremos lo que denominamos como "demandas realistas", que tienen volumen $v(d) \in \{1,\ 2,\ 4\}$. Esta densidad representa casos de uso típicos del mundo real, con slots de 12.5GHz y demandas de 10, 40, y 100Gbps. Adicionalmente contaremos con otras tres posibles densidades que, según el análisis a realizar, pueden ser de interés:

- Densidad Artificial-Baja: $v \in \{1,\ \ldots,\ \overline{s}/3\}$
- Densidad Artificial-Mediana: $v \in \{1,\ \ldots,\ \overline{s}\}$
- Densidad Artificial-Alta: $v \in \{\overline{s}/2,\ \ldots,\ \overline{s}\}$

Los volúmenes artificiales buscan analizar el rendimiento de los modelos a medida que la densidad de las demandas aumenta; además de permitir estudiar cómo escalan los modelos con instancias de mayor tamaño, son de interés por poner en evidencia el comportamiento de los modelos ante instancias fácilmente verificables como factibles o no factibles.

[1] Notemos que al minimizar la cantidad de arcos usados, no habrá ciclos en los caminos y luego cada demanda entra a un nodo a lo sumo una vez.

En todos los casos, el volumen de cada demanda es generado con distribución uniforme en su respectivo conjunto, y la fuente y el destino son generados con distribución uniforme en V.

2.6. Experimentos computacionales

En esta sección estudiaremos la performance de los dos modelos propuestos, evaluando su desempeño al enfrentarse con distintos tipos de instancias. Para determinar el costo computacional que incurrió implementar los distintos esquemas de protección de caminos, compararemos adicionalmente a los modelos propuestos contra DR-AOV, que actuó de punto de partida para los mismos. En todos los casos utilizaremos el software CPLEX como solver para los modelos, y un procesador Intel 12700k. Optamos por un tiempo límite de 900 segundos para la obtención de una solución óptima. Como primer análisis del desempeño de los modelos, estudiaremos su rendimiento al enfrentarse contra demandas de distintas densidades.

2.6.1. DPP

La protección dedicada es un requisito difícil de cumplir, en el sentido de cantidad de recursos de las redes. No solamente pide definir dos caminos disjuntos para cada demanda, sino que los caminos de respaldo de todas las demandas deben ser disjuntos de a pares. Es esperable, entonces, que no se pueda hallar solución alguna para redes chicas y para instancias con demandas de alta densidad debido a la insuficiencia de recursos en las topologías subyacentes. En consecuencia, se espera que las instancias más fácilmente verificables como no factibles sean más fáciles de resolver. Similarmente se espera que el subconjunto de instancias de baja densidad sean más difíciles de resolver, al ser más difícil encontrar un conflicto que prohíbe la factiblidad si la instancia no es factible o al buscar la solución óptima en caso de que sí lo sea.

Para evidenciar el desempeño del modelo DPP bajo distintas densidades de demandas, mostramos en la figura 2.1 los tiempos de ejecución para demandas de densidad baja, mediana, y alta. Estos experimentos fueron realizados sobre la topología 43n-176m-Eurolarge, ya que es la mayor de las topologías encontradas en la literatura y acentúa las diferencias halladas.

En dicha tabla se pueden observar dos resultados principales: por un lado, los mayores saltos en tiempos de ejecución ocurren al aumentar la cantidad de demandas de las instancias, y por otro lado efectivamente las densidades bajas generan las mayores dificultades para el solver, volviéndose más fácil la resolución a medida que aumentan las densidades. En particular se puede ver que los saltos en tiempos de ejecución que ocurren al aumentar la cantidad de demandas es mucho mayor para instancias de densidad baja, mientras que para las densidades normales y altas estos saltos son mucho más atenuados. Esto culmina en un tiempo máximo de 7.22 con $D = 40$, $\overline{s} = 110$, y densidad baja.

Pasando a la suite de instancias construidas a partir de demandas de densidad realista, la figura 2.2 muestra que si bien hay un claro incremento en tiempos de ejecución, como era de esperar, el modelo sigue siendo capaz de resolver las instancias en un plazo de tiempo práctico. En particular algo positivo que se puede ver es que el aumento en tiempos de ejecución al pasar a densidades realistas no es mucho mayor que los saltos vistos en la figura anterior.

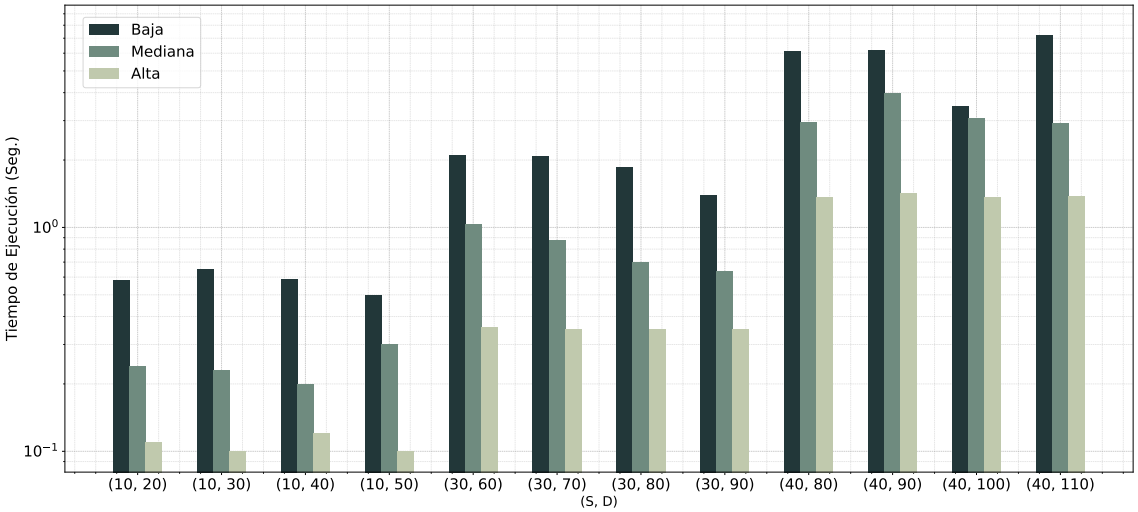

Figura 2.1: *Tiempos de ejecución para el modelo DPP con tres densidades de demandas distintas.*

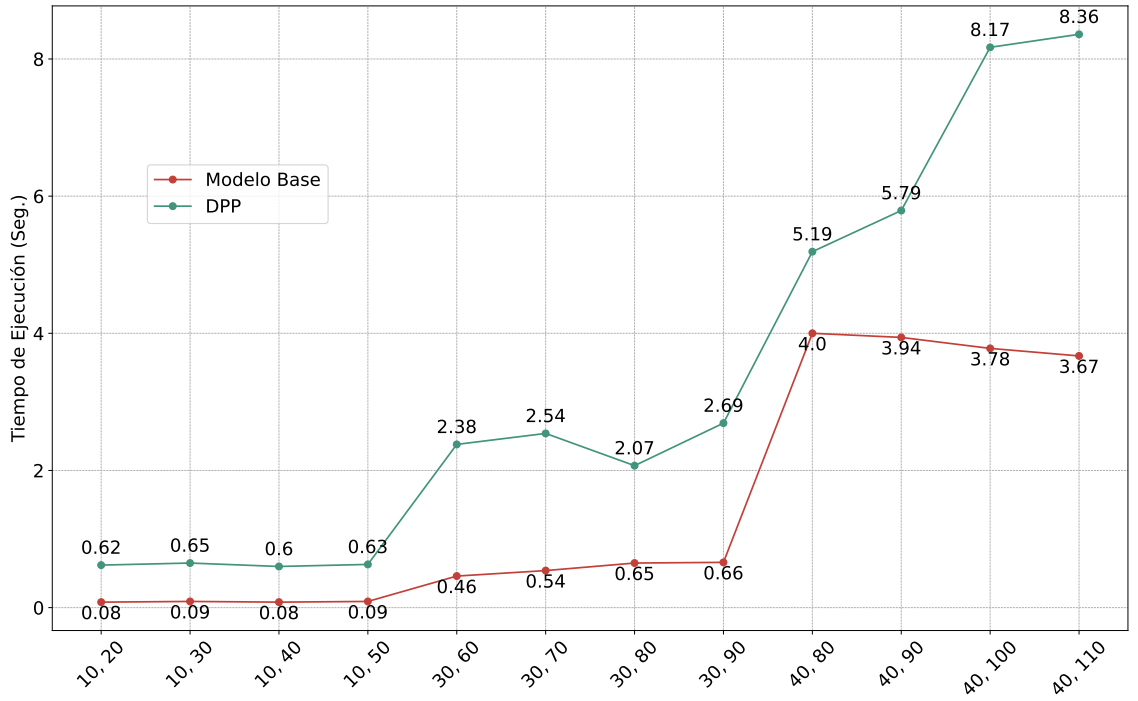

Figura 2.2: *Comparación de tiempos de ejecución entre el modelo base y DPP. Instancia: 43n-176m-EuroLarge, densidad realista.*

2.6.2. SBPP

El segundo modelo propuesto en este trabajo busca implementar la protección de caminos vía recursos de respaldo compartidos. Este problema tiene una lógica inherentemente más complicada que en la protección dedicada; si bien DPP prohíbe por completo el uso compartido de recursos de respaldo, aquí SBPP lo admite siempre y cuando se cumplan ciertas condiciones que debe verificar. Esta verificación implicó en nuestro caso la creación de variables nuevas y restricciones adicionales. En consecuencia, se espera que este modelo sea más difícil de resolver que DPP, requiriendo no solo más tiempo para obtener soluciones óptimas si no también más memoria para lograrlo. La contraparte de esto es que al permitir un uso más comunitario de los recursos la esperanza de factibilidad es mucho mayor, ya que es ahora más probable que las redes den abasto a las demandas definidas.

La figura 2.3 muestra entonces los tiempos de ejecución para el modelo SBPP. Aquí nuevamente utilizamos una instancia con *43n-276m-Eurolarge* como la topología subyacente por ser representativa del patrón general que se observó en la experimentación.

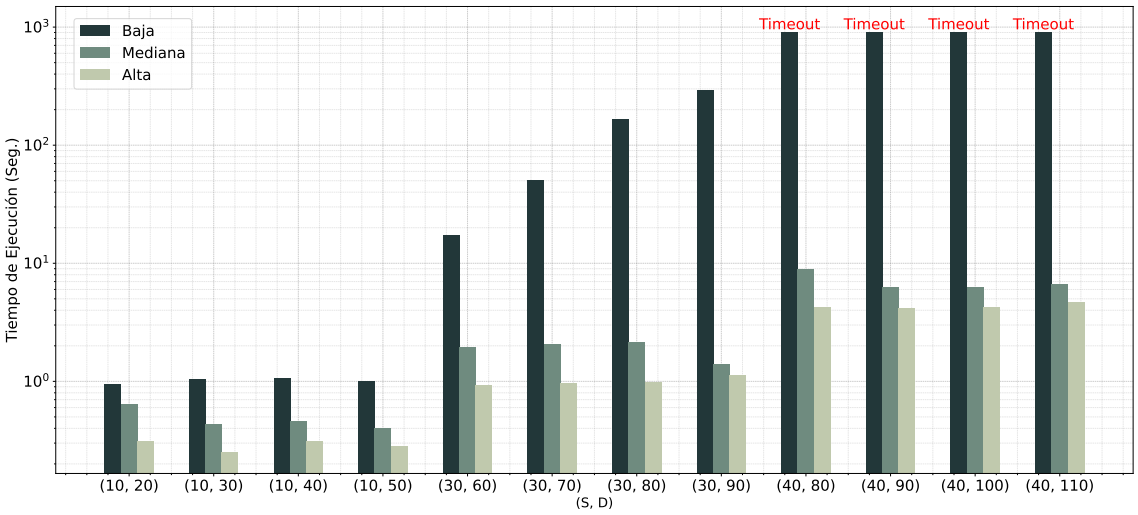

Figura 2.3: *Tiempos de ejecución (en escala logarítmica) para el modelo SBPP con tres densidades de demandas distintas.*

Lo que se puede ver en dicha figura es un patrón muy similar al caso de DPP, con las instancias requiriendo mucho más tiempo de resolución a medida que las densidades bajan. Nuevamente consideramos que esto se debe a que las demandas altan son más fácilmente verificables como no factibles, mientras que en las densidades bajas encontrar un conflicto puede llevar mucho tiempo, y además hay muchas más configuraciones de lightpaths posibles. Cabe destacar que en este modelo la diferencia observada entre cada densidad es mucho más acentuada que para DPP; la brecha en este caso es de varios ordenes de magnitud, y el modelo se queda sin tiempo para las cuatro instancias que buscan establecer lightpaths para 40 demandas.

Para instancias definidas a partir de demandas con volúmenes realistas, el patrón se mantiene. Podemos ver en la figura 2.4 que las cifras de ambos modelos rápidamente divergen, con SBPP alcanzando el timeout para las cuatro instancias de mayor tamaño.

Este problema de performance, que consideramos grave, será algo que buscaremos atacar en los siguientes capítulos de este trabajo.

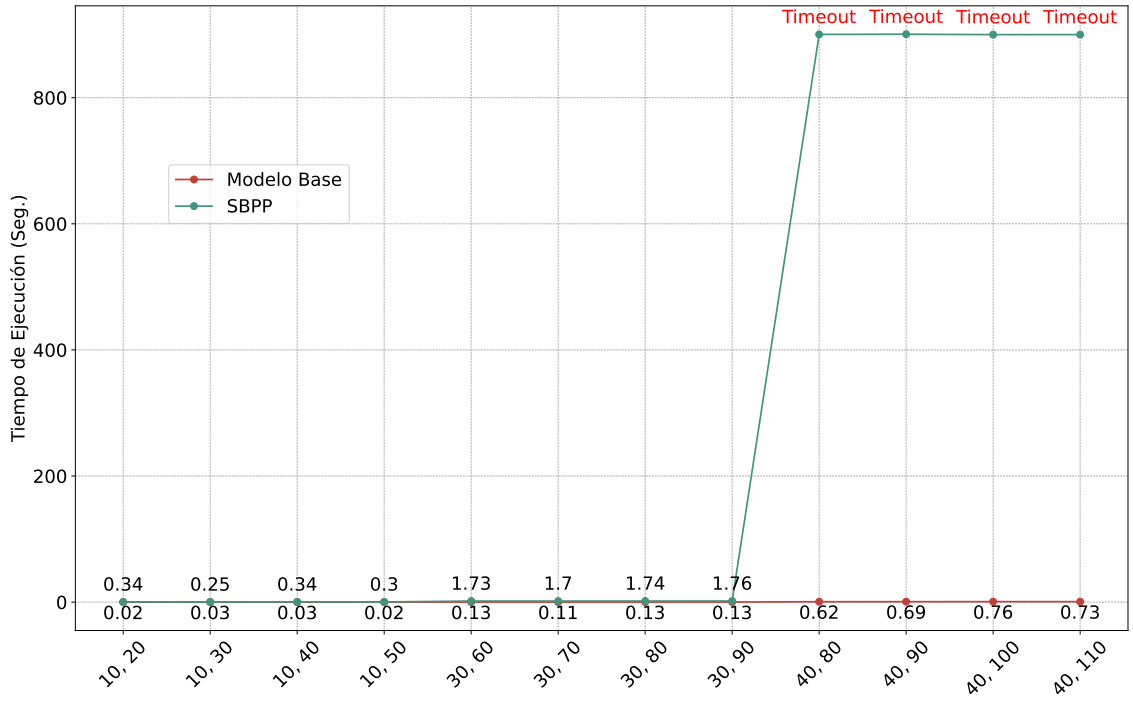

Figura 2.4: *Comparación en tiempos de ejecución entre el modelo base y SBPP. Topología: 43n-176m-EuroLarge. Densidades Realistas.*

2.6.3. Comparación entre modelos

Analizamos también el rendimiento de ambos modelos propuestos en comparación al modelo base. Dado que estos aumentan significativamente el número de demandas y variables, es esperable ver un aumento de tiempo para estos modelos en todos los casos de prueba. La figura 2.5 muestra esta comparación para la topología de mayor tamaño: *43n-176m-EuroLarge*.

Lo que se puede ver, por un lado, es el aumento significativo – pero también esperable – en tiempo de ejecución que presentan ambos modelos propuestos. Ambos aumentan considerablemente la cantidad de variables y restricciones. Por otro lado aquí queda evidenciado el costo adicional incurrido al implementar la protección de caminos vía caminos de respaldo compartidos, que para las instancias con 40 demandas presentan tiempos significativamente peores a DPP. Reiteramos igualmente que estos resultados son esperado ya que SBPP busca resolver un problema con lógica de protección más compleja.

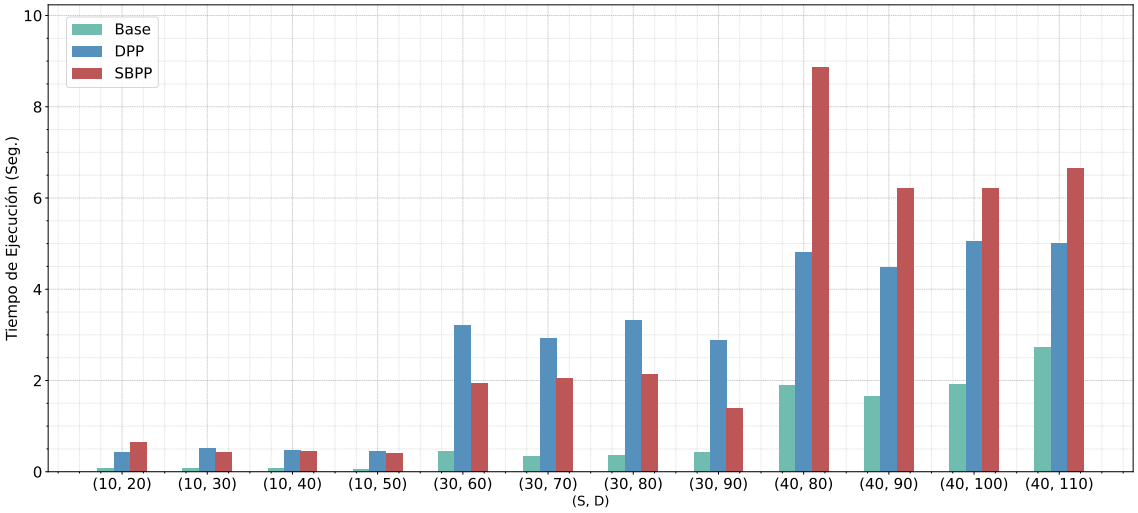

Figura 2.5: *Comparación de tiempos de ejecución entre AOV, DPP, y SBPP. Instancia: 43n-176m-EuroLarge, densidad normal.*

Capítulo 3

Heurísticas primales para el survivable RSA

Durante los experimentos que llevamos a cabo en el capítulo anterior notamos la presencia recurrente de instancias en las cuales si bien el solver encontraba rápidamente una buena cota dual (en muchos casos óptima) para el problema, no encontraba fácilmente una solución inicial factible. Esto nos llevó, naturalmente, a pensar en maneras de ayudar al solver a encontrar rápidamente una solución inicial. Lo que diseñamos fue un modelo PLE auxiliar que resuelve una versión reducida del S-RSA en la que ciertas demandas tienen preasignadas un conjunto limitado de caminos que pueden tomar. La solución obtenida por ese modelo es luego utilizada por el modelo completo como *warm-start*.

En este capítulo entramos más en detalle sobre la heurística propuesta y analizamos su desempeño bajo distintas estrategias de preselección de caminos.

3.1. Trabajo previo

3.1.1. RSA sin protección de caminos

El uso de caminos precomputados para resolver el (S)RSA es extenso y recurrente. Como mencionamos en la introducción de esta parte del trabajo, los primeros modelos para resolver el RSA restringían los caminos admitidos para poder mitigar el crecimiento exponencial de los modelos a medida que crecían las redes, ya que estos utilizaban formulaciones *edge-path* con variables para cada posible camino en la red.

A lo largo de los años se diseñaron heurísticas basadas en la restricción de caminos para las formulaciones *edge-node*, siempre con el objetivo de obtener soluciones de buena calidad (pero no necesariamente óptimas) en un plazo práctico. Una porción de estas heurísticas buscan resolver aproximadamente el problema de ruteo (R), y luego resolver de manera exacta el problema de asignación de espectro (SA). Ejemplos incluyen el trabajo de Wang et al. en 2012 [25] y los trabajos de Christodoulopoulos et al. en 2013 [26] y 2014 [27].

También es común resolver ambas fases del problema heurísticamente, usualmente precomputando los k caminos más cortos para cada demanda y luego aplicando un procedimiento goloso para el SA [14][19][28].

3.1.2. RSA con protección de caminos

Para RSA-DPP, Klinkowski y Walkowiak [29] proponen un procedimiento goloso que asigna lightpaths a demandas en orden decreciente de volumen. Eira et al. [30] diseñan una heurística que precomputa k caminos más cortos para las demandas y busca asignarlos de manera iterativa buscando minimizar la congestión en cada paso. Klinkowski presenta en [31] y [32] metaheurísticas genéticas para este problema. Más recientemente en 2022, Olszewski et al. [33] proponen una heurística para la variante *online* del problema que recorre las demandas utilizando un algoritmo *Max Flow - Min Cost* para asignarles un camino titular y uno de respaldo, asumiendo para simplificar el problema que los slots de cada demanda son los mismos en ambos caminos.

Shao et al. [34] y Tarhan et al. [35] ambos proponen heurísticas para RSA-SBPP que precomputan k caminos con el algoritmo de Yen y luego resuelven SA de manera golosa. Walkwokiak et al. [36] proponen una heurística basada en el criterio *most subcarriers and average longest path first*, que busca asignar lightpaths y slots a las demandas en orden descendiente de volumen, utilizando la longitud de camino esperada para desempates.

Para el multipath RSA, que busca asignar varios caminos a cada demanda y distribuir en ellos la asignación de slots, en Ruan et al. [37] se propone una heurística que utiliza el algoritmo de Bhandari para precomputar caminos y luego resuelve SA de manera golosa, y en Halder et al. [38] se proponen heurísticas golosas y metaheurísticas genéticas para el problema.

En la literatura, la incorporación de modelos PLE para diseñar heurísticas no ha sido significativamente estudiada, ni como herramienta para resolver alguna fase del problema ni para producir soluciones iniciales para un modelo exacto.

3.2. La heurística primal

En esta sección presentaremos la heurística que busca resolver una versión reducida del S-RSA, en la cual ciertas decisiones ya han sido tomadas de antemano. En particular, se precomputará un conjunto reducido de caminos admitidos para ciertas demandas – no necesariamente todas – y se resolverá una variante de los modelos propuestos en donde esas demandas sólo pueden tomar caminos de ese conjunto restringido. La esperanza es que este modelo sea significativamente más fácil de resolver, y que pueda encontrar rápidamente una solución factible para el S-RSA que pueda luego ser usada como warm-start para el modelo completo. De aquí salen algunas consideraciones evidentes:

- El tiempo total que lleva la construcción del conjunto de caminos admitidos más la resolución del modelo auxiliar debería ser menor al tiempo que le lleva al solver encontrar una solución inicial factible. Es decir, el proceso debería poder obtener una solución inicial más rápidamente que el modelo completo.
- El conjunto de caminos admitidos debería ser suficientemente extenso como para que haya esperanza de factibilidad, pero suficientemente acotado como para que el modelo auxiliar resulte significativamente más fácil de resolver.

3.2.1. Cómputo de caminos

Al momento de definir el conjunto de caminos permitidos para las demandas, hay varias decisiones a tomar: cuáles demandas serán las que tengan posibilidades limitadas

de caminos a tomar y, para cada una de estas, qué caminos restringir (los titulares, los de respaldo, o ambos) y cuántas opciones computar.

Lo primero a definir es un procedimiento para generar caminos entre dos nodos, idealmente empezando por el más corto y luego iterativamente generando el siguiente mejor. Un algoritmo conocido para lograr esto es el algoritmo de Yen [39]. Este puede ser usado para obtener una sucesión de los caminos más cortos entre un par de nodos en orden creciente de longitud, y tiene la ventaja de que implementaciones del algoritmo en librerías como *NetworkX* [40] permiten interrumpir esta generación de caminos en cualquier momento. Así, podemos evitar generar todos los caminos entre cada par de nodos, generando en cambio la mínima cantidad que consideremos aceptable. Además, el algoritmo funciona tanto para grafos pesados como no pesados, adaptando según el caso el algoritmo que encuentra caminos más cortos para mejorar el rendimiento.

Una vez que tenemos un mecanismo para generar caminos, debemos decidir si queremos refinar o no el conjunto producido. Por ejemplo, podríamos no imponer ninguna restricción sobre los candidatos, y aceptar cualquier conjunto de caminos. Esto produciría una amplia cantidad de caminos en un tiempo muy corto. Sin embargo, si algún camino tiene una propiedad que impide la factibilidad, es posible que otro camino que comparte muchos de los mismos nodos y arcos también impida la factibilidad por las mismas razones. En otras palabras, este enfoque produce muchos candidatos redundantes, y la diversidad en las opciones se vuelve baja. Podríamos, en el otro extremo, requerir que los caminos candidatos sean disjuntos de a pares; si bien llevaría más tiempo encontrar un conjunto de estos (si es que existen en la red), cuanto mayor sea la diferencia entre cada par de caminos, menor probabilidad hay de que compartan algún defecto que impida la factibilidad. Lo que buscaremos es un punto intermedio que produzca una cantidad útil de caminos con cierta esperanza de diversidad.

Para medir la semejanza entre dos caminos usamos el *índice de Jaccard* [41]. Dados dos caminos c_1 y c_2, interpretados como conjuntos de arcos, su *Jaccard Similarity Score* (JSS) esta dado por:

$$\frac{|c_1 \cap c_2|}{|c_1 \cup c_2|}$$

Experimentamos con diferentes valores del mismo para determinar cuándo dos caminos eran aceptablemente similares y finalmente establecimos un umbral de 0.7, de manera que un camino es admitido a la lista si su JSS con los caminos ya agregados es menor a dicho valor. Este número brinda a la lista de caminos un buen balance entre diversidad y longitud.

Con esto, tenemos las herramientas para definir un esquema *Generate and Test*: generamos para cada demanda los k caminos más cortos entre los nodos fuente y destino con el algoritmo de Yen y nos quedamos con aquellos que sean suficientemente disjuntos a aquellos ya seleccionados.

3.2.2. Modelos auxiliares

A continuación detallamos los modelos que buscan resolver una versión reducida del problema, para obtener soluciones iniciales al problema completo rápidamente. Definimos primero la notación nueva asociada a estos modelos:

Constantes:

C	El conjunto de caminos precomputados para todas las demandas
C_d	El subconjunto de C de los caminos precomputados para la demanda d

Variables:

zm_{dc}	igual a 1 si la demanda d utiliza el camino c como camino titular
zb_{dc}	igual a 1 si la demanda d utiliza el camino c como camino de respaldo

Modelando la restricción de caminos titulares

Supongamos que ya hemos realizado un precómputo de caminos candidatos, de manera que las demandas seleccionadas tienen asignadas un conjunto de caminos posibles entre los cuales debe elegir. Para obtener una solución al S-RSA donde cada demanda d pueda utilizar solamente caminos de C_d como camino titular, podemos agregar a los modelos DPP y SBPP las siguientes restricciones:

$$\sum_{c \in C_d} zm_{dc} = 1 \qquad \forall d \in D \tag{3.1}$$

$$\sum_{\substack{c \in C_d / \\ e \in c}} zm_{dc} = ym_{de} \qquad \forall d \in D, e \in E \tag{3.2}$$

$$zm_{dc} \in \{0, 1\} \qquad \forall d \in D, c \in C \tag{3.3}$$

La primera restricción elige exactamente un camino titular a tomar para cada demanda. La segunda vincula la elección de un camino c para la demanda d con la activación de ym_{de} para cada uno de los arcos de c.

Notemos que agregar estas restricciones adicionales es suficiente para obtener los modelos reducidos tanto en DPP como en SBPP; la restricción (3.2) de arriba vincula lógicamente la elección de caminos con la activación de variables ym, y los modelos ya imponen restricciones en las relaciones que estas últimas variables pueden tener entre sí. De esta manera, cualquier elección de caminos inválida en algún modelo implicará valores de ym que violen las restricciones y será descartada. Asimismo, una demanda d no podrá seleccionar un arco e en su camino titular – es decir, activar ym_{de} – si dentro de los caminos candidatos no hay algún camino c que pase por e.

Modelando la restricción de caminos titulares y de respaldo

El modelo reducido que precomputa un conjunto de opciones para ambos tipos de caminos de cada demanda es similar al modelo anterior. En este caso, agregamos a DPP y SBPP las siguientes restricciones:

$$\sum_{c \in C_d} zm_{dc} = 1 \qquad \forall d \in D \tag{3.4}$$

$$\sum_{c \in C_d} zb_{dc} = 1 \qquad \forall d \in D \tag{3.5}$$

$$\sum_{\substack{c \in C_d / \\ e \in c}} zm_{dc} = ym_{de} \qquad \forall d \in D, e \in E \tag{3.6}$$

$$\sum_{\substack{c \in C_d / \\ e \in c}} zb_{dc} = yb_{de} \qquad \forall d \in D, e \in E \tag{3.7}$$

$$zm_{dc},\ zb_{dc} \in \{0, 1\} \qquad \forall d \in D, c \in C \tag{3.8}$$

Nuevamente, se define un camino para cada demanda d y se vincula lógicamente dicho camino con cada variable y asociada a d y a los arcos del camino. Recalcamos que no es necesario definir lógica para prevenir conflictos entre los caminos titulares y de respaldo, pues son las variables ym e yb quienes se encargan de eso implícitamente.

3.3. Calidad de las soluciones heurísticas

El objetivo de los modelos auxiliares es encontrar rápidamente una solución factible al S-RSA, para que los modelos completos la usen de *warm-start*. En el mejor de los casos, los modelos auxiliares no sólo producirían soluciones rápidamente, sino que estas serían de alta calidad.

Es importante notar que el principal factor que influye en el tiempo de obtención y en la calidad de las soluciones es el conjunto de caminos predeterminados que computamos, y que el tamaño determina cuál objetivo se prioriza. Cuanto más acotado sea el conjunto de caminos, más simple será el problema a resolver, y se espera que el modelo auxiliar encuentre rápidamente una solución. Hay que tener cuidado, sin embargo, de no restringir excesivamente el problema a tal punto que se vuelva no factible. Por otro lado, darle a cada demanda un conjunto de caminos más extenso – si bien dificulta el problema y empeora los tiempos de resolución – puede llevar a mejores soluciones, y se espera que partir de una solución de mejor calidad aumente la eficiencia de los modelos completos.

Esta sección se dedicará a explorar el desempeño de los modelos bajo distintas longitudes para los caminos predeterminados. Estudiamos los tiempos de ejecución totales, incluyendo el cómputo de los caminos, la resolución del modelo auxiliar y la solución del modelo completo con el warm-start obtenido. También estudiamos la calidad de las soluciones provistas por el modelo auxiliar.

3.3.1. DPP

3.3.1.1. Restricción de caminos titulares

k = 20

Estudiamos primero la heurística que restringe solamente caminos titulares. La figura 3.1 muestra el *optimality gap* entre las soluciones devueltas por el modelo reducido contra la solución óptima.

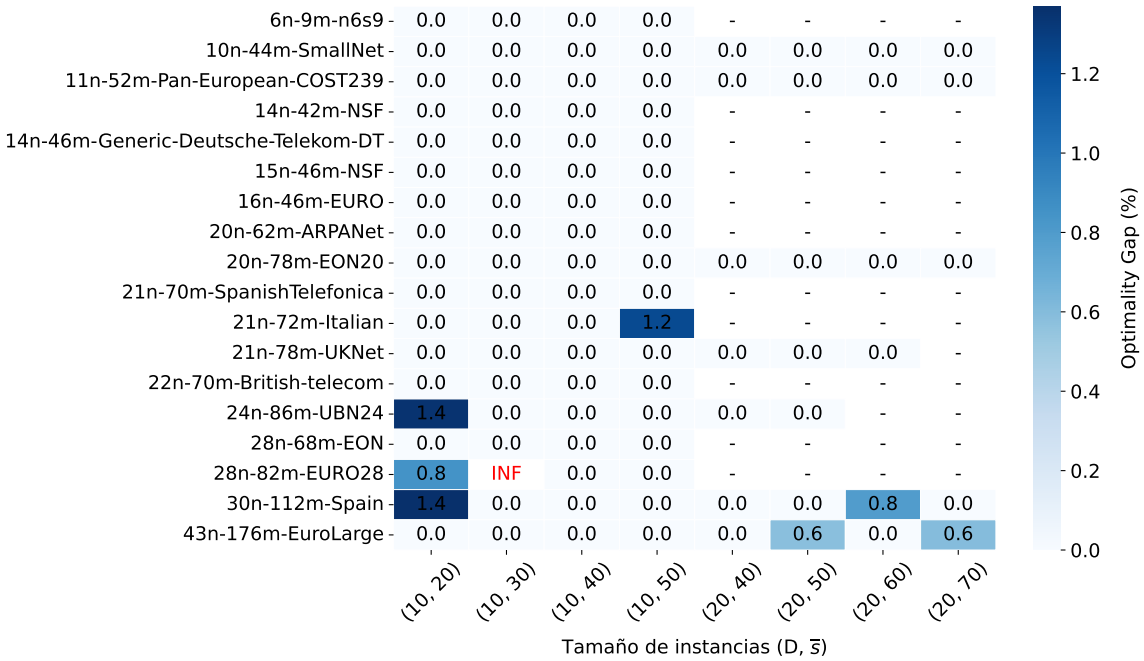

Figura 3.1: *Optimality gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. k=20*

Es fácil reconocer la alta calidad de las soluciones provistas por el modelo reducido; en la gran mayoría de las instancias, la solución hallada es la óptima. Esto no solo sugiere que las soluciones serán muy útiles como warm starts del modelo completo, posiblemente reduciendo significativamente el tiempo de cómputo, sino que también esta estrategia sirve por sí sola como heurística con soluciones de alta calidad esperada.

La figura 3.2 muestra los tiempos de ejecución del pipeline completo contra el modelo base. Como era esperable, los tiempos de ejecución empeoran para las instancias chicas ya que como los tiempos de resolución para estas instancias no son altos en primer lugar, invertir el tiempo en un pipeline entero de obtención de soluciones iniciales y su posterior utilización como warm-start no vale la pena. No obstante, los aumentos en estas instancias corresponden a pocas décimas de segundos.

Por otro lado, a medida que las instancias aumentan en complejidad – ya sea por el tamaño de la red o por la cantidad de demandas y slots – ejecutar el pipeline completo definitivamente se vuelve una inversión de tiempo prudente en la mayoría de los casos. Cabe destacar que en la mayoría de las instancias grandes en las cuales el pipeline presenta peores tiempos de ejecución, lo que le cuesta al solver es obtener una buena cota dual. Es razonable que en estos casos los warm-starts no ayuden, ya que por más rápidamente que se obtenga una solución inicial factible, la dificultad en encontrar cotas duales es el cuello de botella.

k = 10

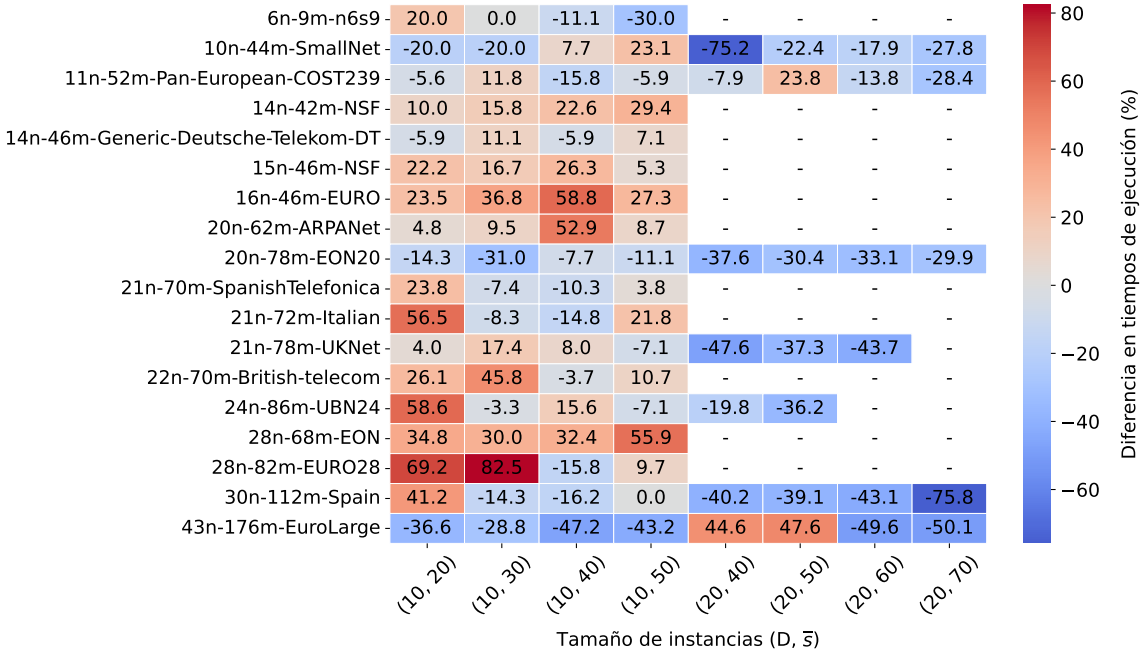

Figura 3.2: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. k=20*

Con el fin de identificar un número óptimo de caminos a fijar, estudiamos ahora la calidad y utilidad de las soluciones del modelo reducido cuando $k = 10$. Reducir el número de caminos computados produciría un modelo inicial más fácil de resolver, pero que bien puede devolver warm-starts de peor calidad y aumentar la cantidad de casos en los cuales no se devuelve solución alguna.

Se observa en la figura 3.3 que las soluciones halladas siguen siendo de muy buena calidad, hallando en la gran mayoría de los casos la solución óptima de las instancias. Por otro lado se puede ver que aumenta levemente la cantidad de instancias en las que la solución hallada no es óptima, y también es importante notar que efectivamente hubo un aumento en casos en donde el modelo no es factible.

Más allá de la leve pérdida de calidad de las soluciones iniciales, la figura 3.4 muestra resultados positivos en términos de los tiempos de resolución. Por un lado, las instancias grandes siguen presentando mejoras significativas en sus tiempos de ejecución. Por otro lado, al ser menor el tiempo de obtención de los warm-starts, ahora también vale la pena llevar a cabo el pipeline para instancias de tamaño mediano y pequeño en la gran mayoría de los casos.

k = 5

Con el fin de seguir explorando la relación entre la cantidad de caminos precomputados y el desempeño de la heurística para hallar un punto medio ideal, estudiamos ahora el desempeño de la heurística al computar cinco opciones de demandas para sus caminos titulares.

Los optimality gaps que muestra la figura 3.5 continúan la tendencia hacia abajo en calidad que se veía a medida que se reducen los caminos computados para cada demanda.

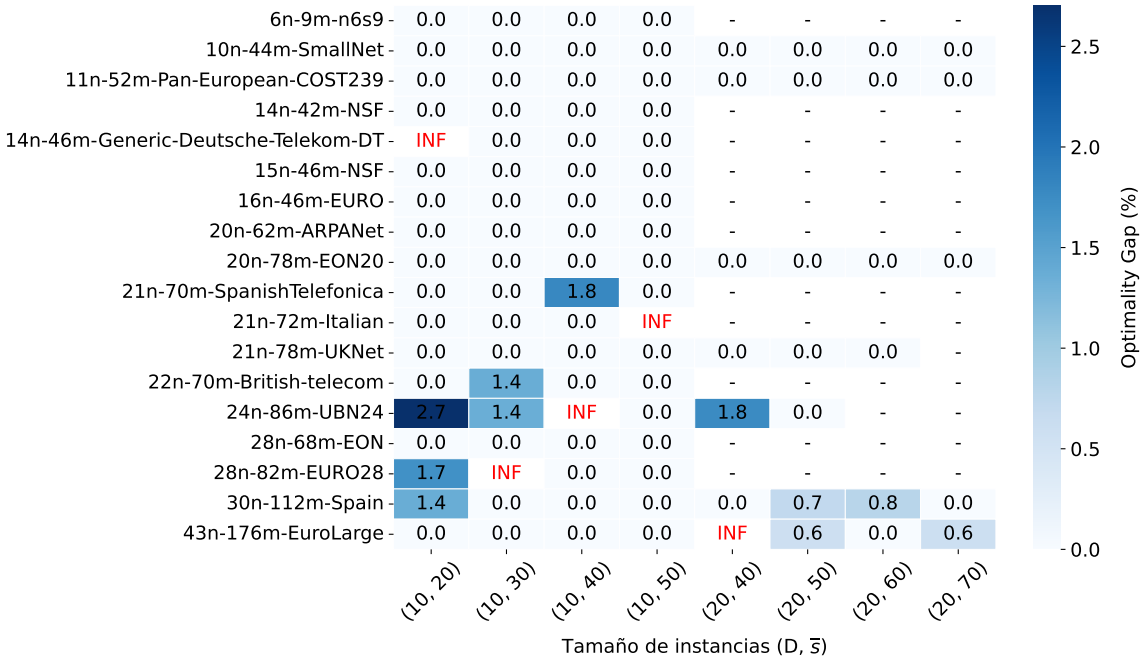

Figura 3.3: *Optimality gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. k=10*

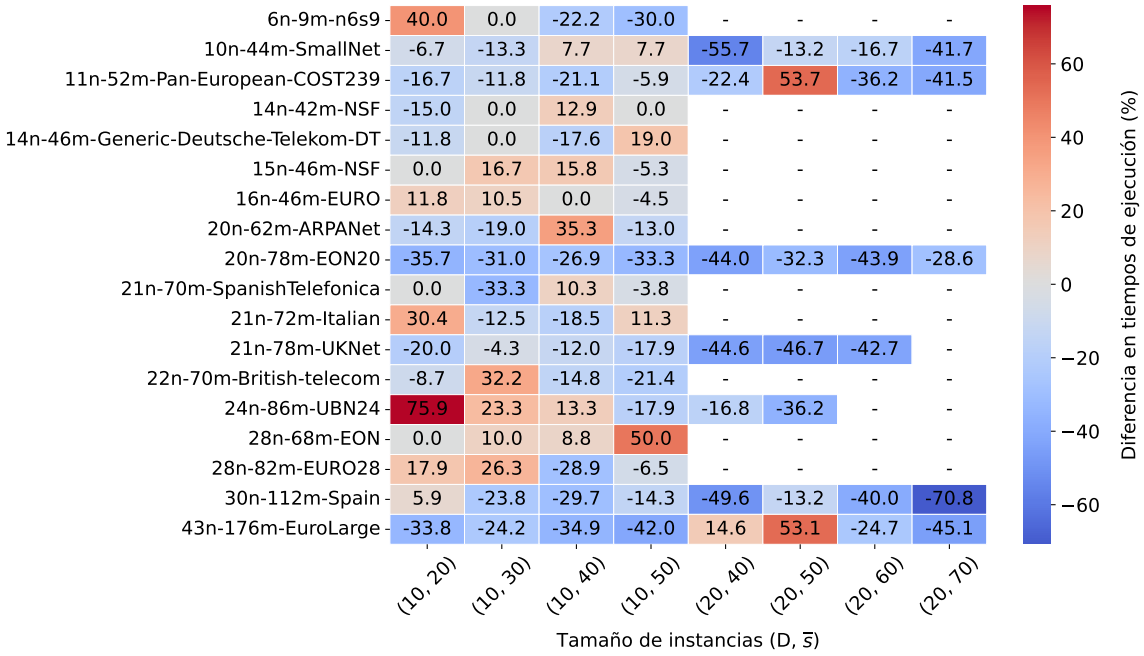

Figura 3.4: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. k=10*

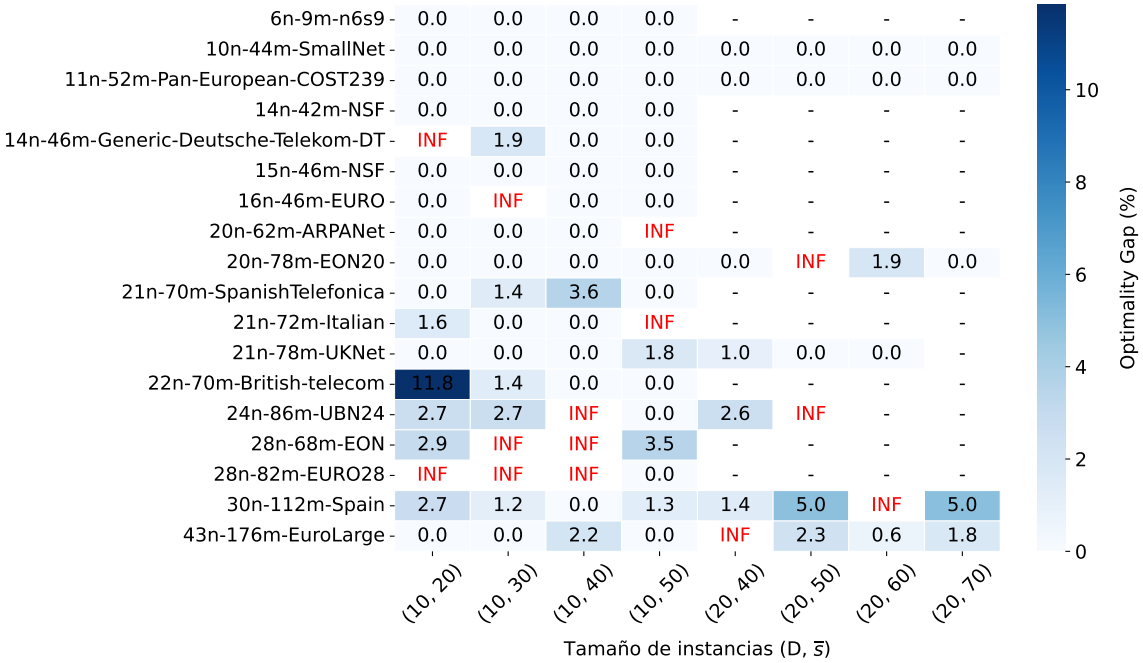

Figura 3.5: *Optimality gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. k=5*

En particular, se nota un deterioro notable en las soluciones halladas para instancias grandes.

La figura 3.6 muestra por primera vez peores tiempos de ejecución para instancias grandes que el modelo DPP base. Dado que naturalmente las instancias grandes son las que más tardan en ser resueltas, no es sorprendente que los warm-starts de relativamente baja calidad provistos para ellas afecten negativamente los tiempos de ejecución. Por otro lado, hay por primera vez una mejora general en los tiempos de ejecución para las instancias chicas; con $k = 5$, el cómputo de caminos y la resolución del modelo auxiliar finalmente son lo suficientemente rápido para vencer el bajo tiempo de resolución del modelo DPP base.

3.3.1.2. Restricción de caminos titulares y de respaldo

Consideramos que también es de interés estudiar el desempeño del modelo reducido al limitar las opciones de caminos que las demandas pueden tomar tanto en sus caminos titulares como de respaldo. Esto nos permitirá prescindir de cierta esperanza de factibilidad a cambio de un menor tiempo esperado de obtención de warm-starts. Es decir, esperamos que los optimality gaps en estos casos sean peores que aquellos obtenidos en la sección anterior, pero idealmente conducirán en la mayoría de los casos a una reducción en tiempos de ejecución.

La experimentación es muy similar a la realizada en la sección anterior, ya que aquí también estudiaremos los optimality gaps y tiempos de ejecución según la cantidad de caminos precomputados. Por cuestiones de experiencia lectora presentaremos aquí los mejores resultados obtenidos en esta tanda de experimentación; el resto de los resultados estarán disponibles en el anexo.

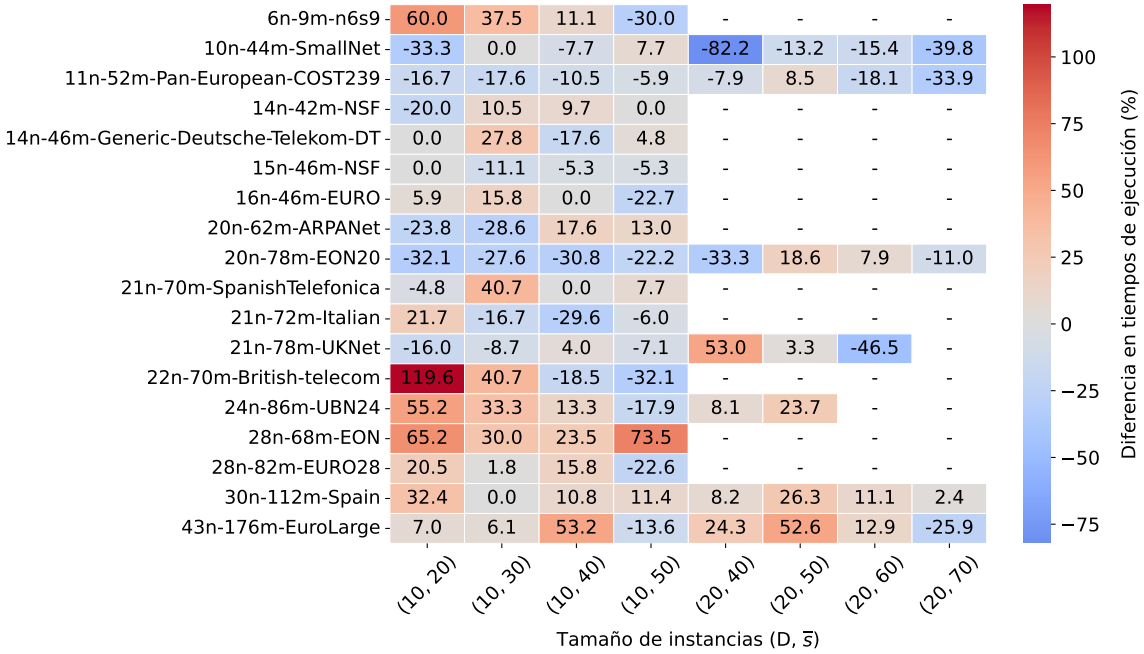

Figura 3.6: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. k=5*

k = 10

Las figuras 3.7 y 3.8 muestran respectivamente los optimality gaps y tiempos de ejecución cuando se precomputan 10 caminos para cada demanda, entre los cuales deberá elegir uno de titular y uno de respaldo. Lo que se puede ver, por un lado, es que una cantidad considerable de instancias no alcanzan el óptimo del problema completo. Esto es esperable dado que si bien hay 10 caminos candidatos, ahora se deben seleccionar dos de estos y además deben ser disjuntos. No obstante, los resultados en términos del tiempo total de ejecución son positivos: con esta configuración se alcanza un buen desempeño en instancias de todos los tamaños, logrando reducir los tiempos de ejecución en la mayoría de instancias tanto chicas como medianas y grandes.

3.3.2. SBPP

3.3.2.1. Restricción de caminos titulares

A continuación detallamos los resultados obtenidos al aplicar las heurísticas presentadas al modelo SBPP. En particular haremos comentario sobre el caso $k = 5$, que (a pesar de no poder obtener soluciones factibles en varios casos) produjo los mayores ahorros de tiempo; el análisis hecho para otros valores de k es análogo al realizado en la sección anterior para DPP, y por cuestiones de brevedad nuevamente delegamos el resto de los resultados al anexo.

k = 5

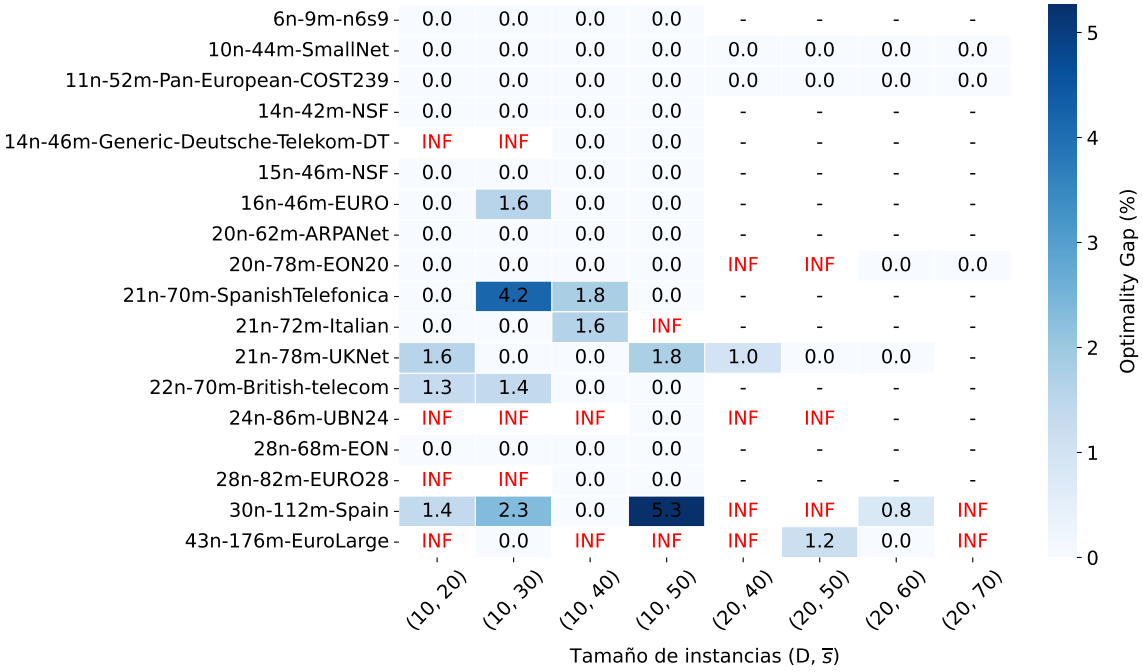

Figura 3.7: *Optimality gap entre soluciones de modelo reducido (fijando caminos titulares y de respaldo) y soluciones exactas, para un conjunto de instancias factibles. Modelo: DPP Caminos a fijar: Titulares y de Respaldo. k=10*

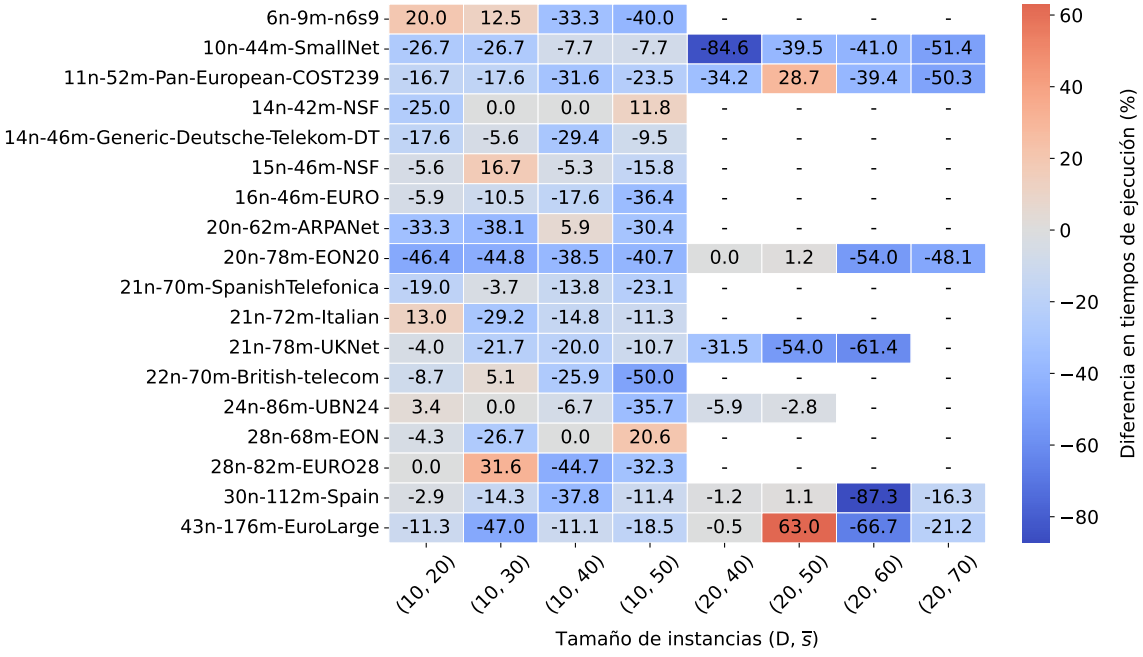

Figura 3.8: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística (fijando caminos titulares y de respaldo) + modelo completo con el warm-start provisto. Modelo: DPP Caminos a fijar: Titulares y de Respaldo. k=10*

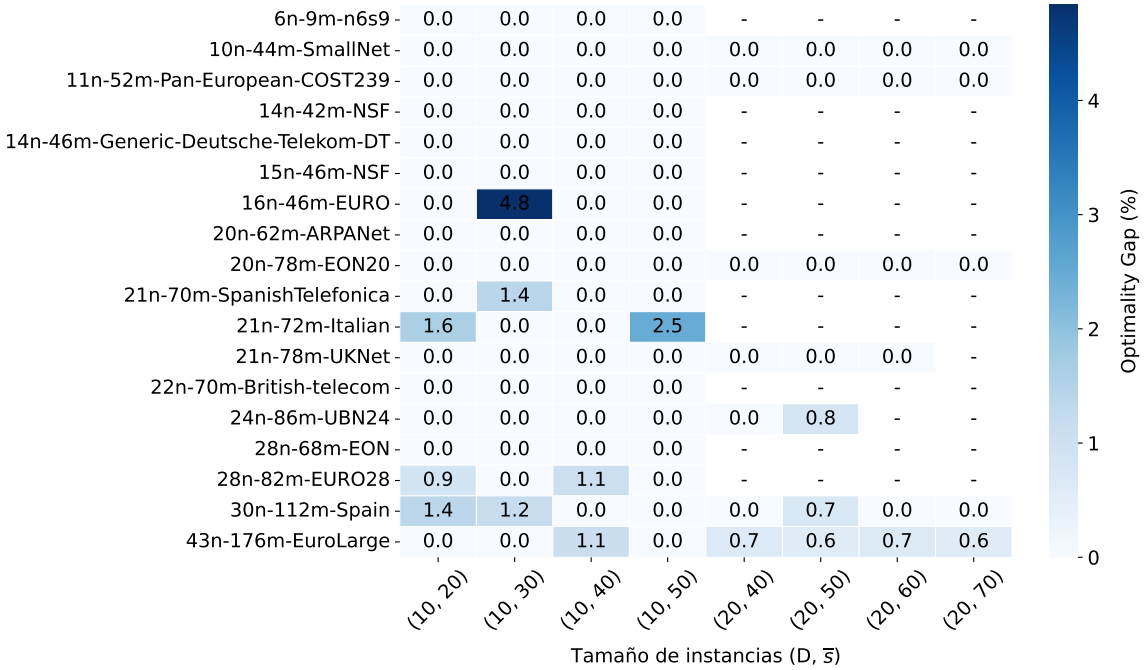

Figura 3.9: *Optimality gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. k=5*

Lo que se puede ver en la figura 3.9 es que nuevamente fijar solo cinco caminos para cada demanda trae complicaciones con la factibilidad en muchos casos, que se ven exacerbados a medida que crecen las instancias tanto en tamaño de topología como en cantidad de demandas.

La figura 3.10, que detalla la diferencia en tiempos de ejecución (ya sea para mejor o peor) entre el modelo SBPP base y el tiempo total de la heurística y la posterior resolución del modelo completo con el warm-start obtenido, muestra resultados que no son malos pero que tampoco destacaron. Lo que presentan es un escenario típico en esta heurística: para muchas de las instancias grandes se obtienen reducciones significativas en tiempos de ejecución, pero en el resto de los casos (particularmente para instancias chicas) la heurística no mejora los tiempos de ejecución respecto del modelo base.

3.3.2.2. Restricción de caminos titulares y de respaldo

La última heurística propuesta en este trabajo fija caminos titulares y de respaldo para las demandas, en un ruteo con protección de caminos vía recursos de respaldo compartidos. Presentaremos en esta sección los mejores resultados obtenidos para este caso y detallaremos el resto de los resultados en el anexo. Estudiando los tiempos de ejecución dados por valores $k = 10,\ 20,$ y 40, seleccionamos $k = 10$ como el parámetro óptimo.

k = 10

Por el lado de la función objetivo, podemos ver en la figura 3.11 que el rendimiento no es muy destacable; a medida que crecen las instancias hay varias de ellas para las cuales el modelo reducido no es capaz de obtener soluciones factibles. Sin embargo, los tiempos

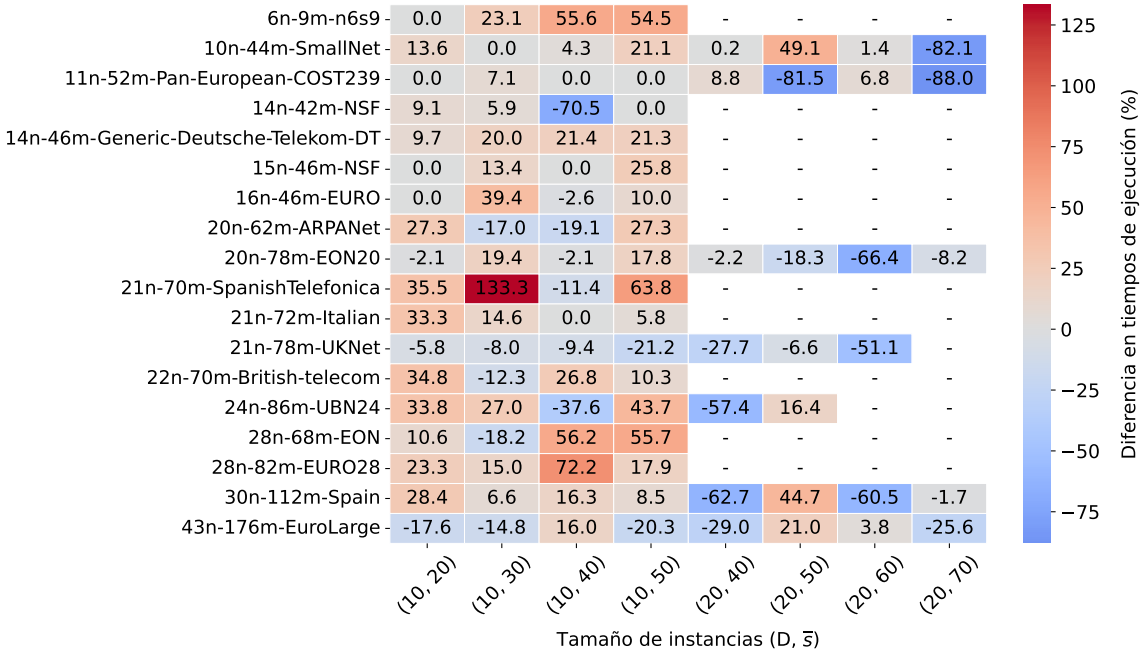

Figura 3.10: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. k=5*

de ejecución exhibidos en la figura 3.12 muestran que aún así los warm starts provistos por el modelo reducido terminan acelerando la obtención de soluciones óptimas al modelo completo en la gran mayoría de las instancias. Estos resultados muestran una reducción global de los tiempos de resolución, mejorándolos para instancias de todos los tamaños.

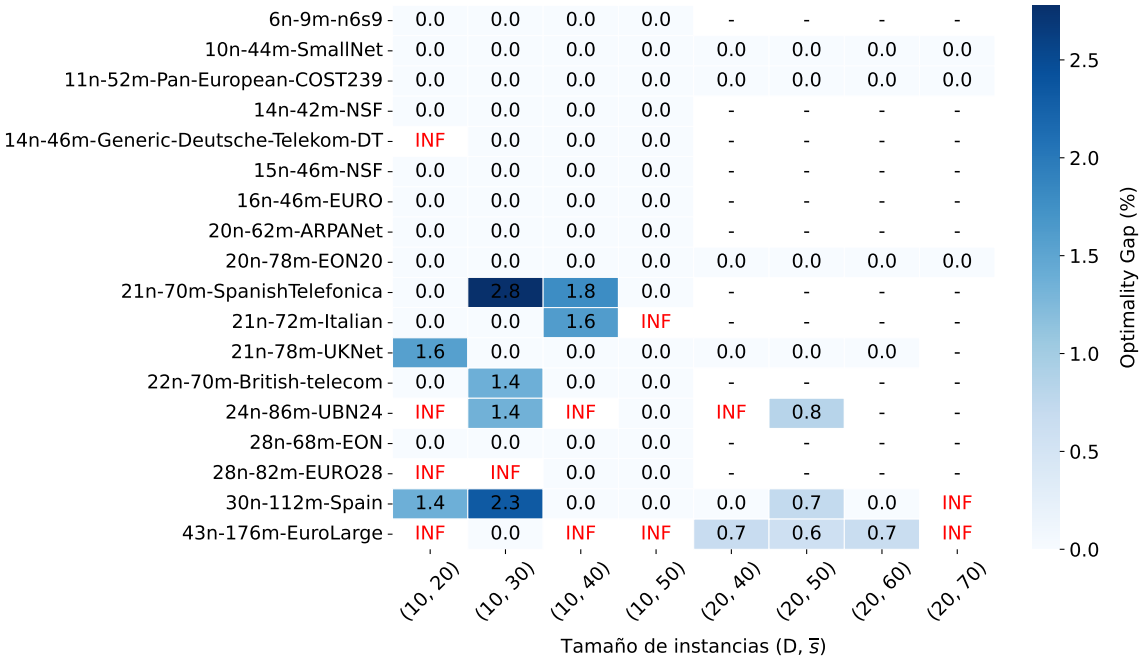

Figura 3.11: *Optimality gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. Modelo: SBPP. Caminos a fijar: titulares y de respaldo. k=10*

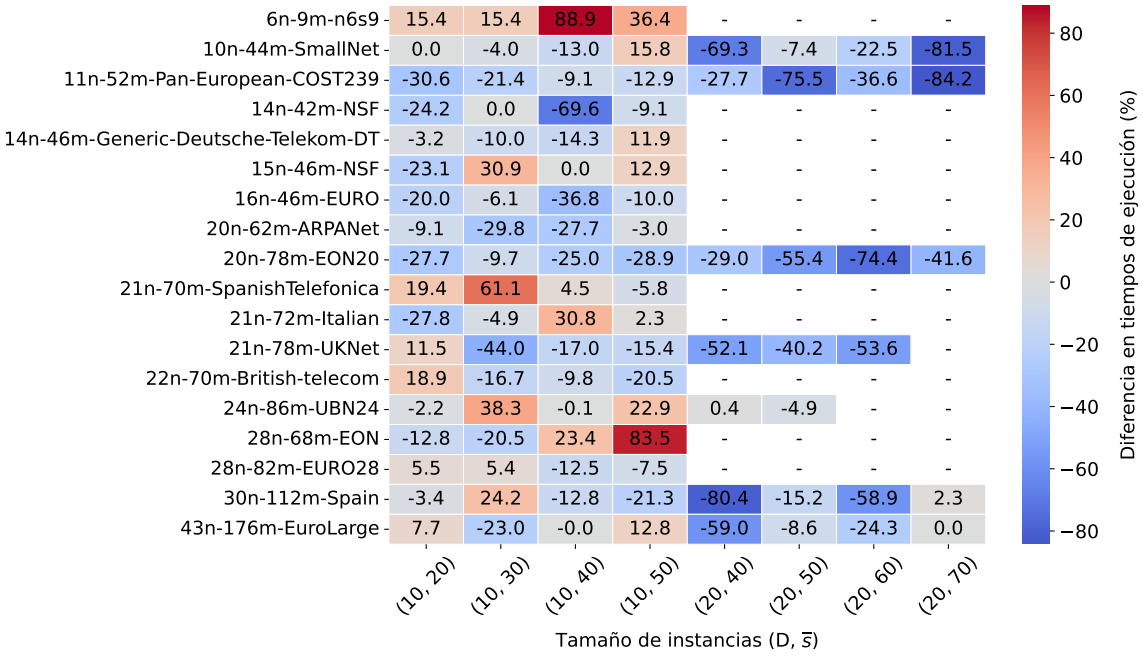

Figura 3.12: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. Modelo: SBPP. Caminos a fijar: titulares y de respaldo. k=10*

Parte II

Capítulo 4

Esquemas de descomposición para RSA

En la primera parte de este trabajo diseñamos varios modelos para el S-RSA y estudiamos su rendimiento usando CPLEX como solver de caja negra. Si bien es esperable que el solver implemente una variedad de optimizaciones "detrás de escena", esta estrategia de resolución es genérica y no necesariamente aprovecha al máximo la estructura particular del problema.

Así, en esta segunda parte del trabajo ponemos el foco tanto en RSA como S-RSA y buscamos diseñar estrategias alternativas para resolver los modelos existentes, que utilicen optimizaciones ad-hoc para acelerar la obtención de soluciones o la detección de su ausencia.

4.1. Descomposición combinatoria de Benders

El problema de Ruteo y Asignación de Espectro puede, como el nombre sugiere, interpretarse como un problema en dos partes: el ruteo de un conjunto de demandas a lo largo de la red, y la asignación de slots a cada una de estas demandas. Si bien estas dos fases no son completamente independientes –la factibilidad de cada fase depende de la configuración actual de la otra– el problema tiene una estructura suficientemente disjunta como para explorar esquemas de descomposición para los modelos. Estos esquemas buscan separar un modelo difícil de resolver en modelos menos complejos, y ponerlos en un estado de constante comunicación, en el cual se ayudan mutuamente a lograr un objetivo común.

Un ejemplo clásico en la literatura de estos esquemas es la descomposición de Benders [42]. En términos generales, este busca resolver un problema descomponiéndolo en dos: un problema maestro y un subproblema auxiliar dual que busca orientar al problema maestro en su búsqueda de una solución óptima al problema general. Bajo este marco, el modelo maestro busca resolver una de las fases del problema, y luego el modelo auxiliar busca completar la resolución del problema sujeto a las condiciones impuestas por la solución maestra hallada. Como el problema maestro no tiene contexto del problema general, es posible que la primera solución parcial que produzca no le sirva al modelo auxiliar. El rol de este segundo modelo, entonces, es producir cortes que al agregarse al modelo maestro lo robustecen, otorgándole información sobre por qué las soluciones encontradas hasta el momento no son óptimas para el problema general. Cuando ambos modelos son comparativamente fáciles de resolver por separado, y la cantidad de iteraciones realizadas

de este proceso es baja, esta descomposición optimiza significativamente la obtención de soluciones.

Por alivianar enormemente la carga computacional – en términos tanto de tiempos de ejecución como de consumo de memoria – incurrida por modelos complejos y en cambio permitiendo resolver un modelo mucho más simple a la vez, este esquema ha sido extensamente aplicado a problemas de gran escala que surgen en logística [43], planificación de inversiones [44], medicina [45], entre muchos otros.

El esquema clásico propuesto por Jacques Benders ha sido generalizado / extendido con mucho éxito para aplicarlo a un rango más amplio de problemas, permitiendo por ejemplo que el subproblema sea un problema de programación no lineal [46] o directamente un problema de optimización de cualquier tipo [47]. Además, en Fischetti et al. [48] los autores proponen lo que se suelen llamar *Cortes Combinatorios de Benders*, en donde el problema auxiliar no necesariamente tiene función objetivo, y su rol principal es generar cortes por factibilidad que prohíben ciertas combinaciones de variables en el problema maestro. En este esquema, se busca aprovechar la estructura del problema para obtener cortes minimalmente infactibles en la esperanza de que estos cortes resulten fuertes y aceleren los algoritmos. Los algoritmos de descomposición que propondremos en este trabajo se inspiran en estas variantes.

4.2. Aplicación al RSA

Para descomponer el RSA en dos fases y definir un algoritmo que implemente el esquema descripto en la sección anterior, decidimos tratar al problema de ruteo como el problema maestro y el problema de asignación de espectro como problema auxiliar. De este modo, el modelo maestro deberá encontrar un ruteo para el conjunto de demandas dado, y el modelo auxiliar buscará una configuración de slots para estas demandas tal que, con el ruteo producido, no haya superposición de slots entre demandas cuyos caminos se cruzan.

Notemos que como cada modelo debe ser agnóstico a las variables y restricciones del otro modelo, debemos de alguna forma “traducir” la solución entregada por el modelo principal para que el modelo auxiliar pueda igualmente resolver la asignación de espectro sin acceso a las variables de ruteo. Esta traducción se realizará construyendo un grafo de interferencia asociado al ruteo hallado que indica cuáles demandas comparten arcos en sus caminos asignados, y el modelo auxiliar será resuelto recibiendo este grafo como dato. Este proceso es descripto en más detalle en las siguientes secciones.

Si el modelo auxiliar determina que no existe una configuración válida de los slots para un ruteo dado, nuestro algoritmo identificará qué propiedades del ruteo no permiten la factibilidad y agregará al modelo maestro cortes que prohíben estas situaciones. Notemos que bajo la versión de RSA que busca minimizar la cantidad de arcos usados, si el ruteo hallado admite una configuración de slots válida entonces este ruteo junto a dicha configuración de slots define una solución óptima al problema completo. Por otro lado, si el modelo maestro no produce una solución factible sabremos que nuestra instancia no tiene solución.

El diagrama de la figura 4.1 visualiza el flujo del algoritmo descripto.

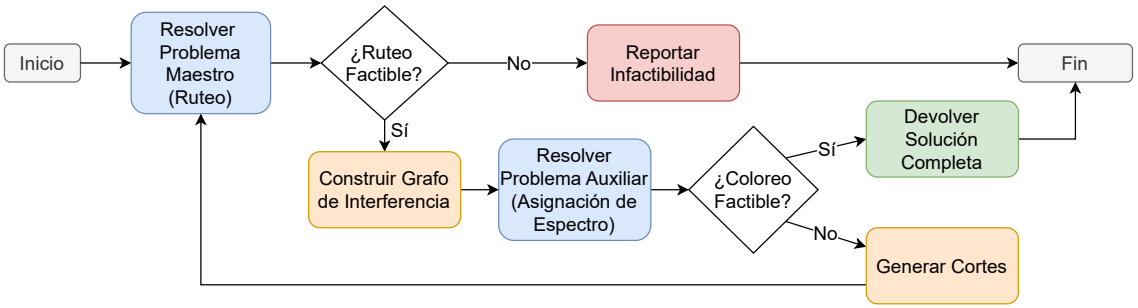

Figura 4.1: *Diagrama de flujo del algoritmo iterativo propuesto para resolver el RSA. Los nodos azules indican la resolución de un modelo de programación entera, mientras que los nodos naranjas indican procesamiento adicional realizado sobre las soluciones obtenidas.*

4.3. Modelo maestro

A continuación presentamos concretamente el modelo que usaremos para resolver el problema principal de la descomposición, que en nuestro caso es el problema de ruteo. La construcción de este modelo es sencilla: se remueven del modelo original las variables y restricciones relacionadas a los slots, preservando solamente aquello relacionado a la obtención de un ruteo para cada demanda. Así, el modelo resultante para esta fase es:

$$\min \quad \sum_{d \in D} \sum_{e \in E} y_{de}$$

$$\text{s.a.} \sum_{e \in \delta^+(v)} y_{de} - \sum_{e \in \delta^-(v)} y_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{otherwise} \end{cases}, \qquad \forall\, d \in D, v \in V \tag{4.1}$$

$$\sum_{d \,\in\, D} v(d)\, y_{de} \leq S, \qquad \forall\, e \in E \tag{4.2}$$

$$y_{de} \in \{0, 1\}, \qquad \forall\, d \in D, e \in E \tag{4.3}$$

$$l_d, r_d \in \mathbb{Z}, \qquad \forall\, d \in D \tag{4.4}$$

Notemos que la restricción (4.2) es nueva. Esta pide que la suma de los volúmenes de las demandas que pasan por cada arco no superen la capacidad del carrier. En otras palabras, pide que ningún enlace quede saturado. Esta restricción se agrega al modelo para que el punto de partida de las soluciones halladas sea mínimamente astuto; si no estuviera esta restricción, el primer ruteo hallado sería simplemente el camino más corto para cada demanda, sin ninguna consideración por el hecho de que estos caminos tienen que convivir en una red de capacidad limitada. Ese modelo, por supuesto, no conduciría de entrada a un ruteo factible para el problema en general, y requeriría la adición de varias desigualdades adicionales – producidas en cada iteración del algoritmo – para empezar a producir ruteos plausibles.

4.4. Modelo auxiliar

El modelo auxiliar que usaremos es, a grandes rasgos, el modelo original sin las variables y restricciones relacionadas a la lógica de ruteo. Sin embargo, en esta fase debemos primero

realizar un procesamiento de la solución entregada por el problema maestro antes de poder encontrar una asignación de slots para ella. Esto se debe a que la solución está expresada en términos de las variables de ruteo, las cuales no incumben al modelo auxiliar.

Recordemos que para el modelo que resuelve SA el objetivo principal es asegurar que si los caminos asignados a dos demandas se cruzan, estas dos demandas viajen por slots distintos. Más allá de eso, el resto de la lógica del modelo se preocupa por mantener los slots de cada demanda dentro de un rango especificado, y esto no depende del ruteo. De esta manera podemos pensar que en la segunda fase de la descomposición, la única información relevante sobre el ruteo hallado tiene que ver con cuáles demandas comparten arcos, ya que son estas las que imponen restricciones sobre la asignación de slots.

Con esto en mente, podemos construir un grafo de interferencia M con un nodo por demanda, en donde dos nodos son adyacentes si las rutas de las demandas asociadas comparten enlaces en la red. La figura 4.2 ilustra esta construcción.

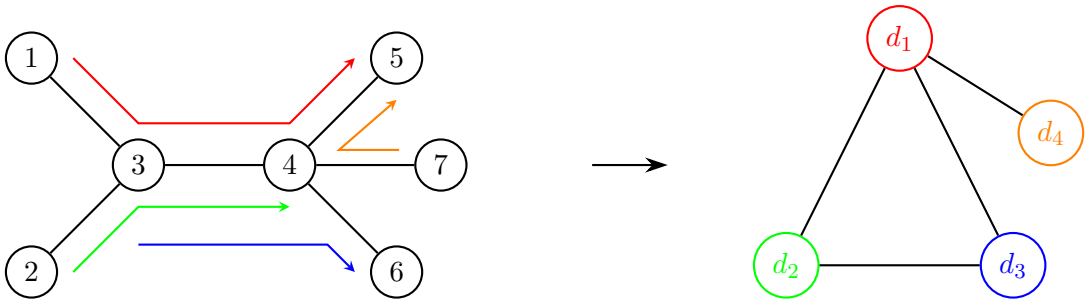

Figura 4.2: *El ruteo de cuatro demandas a través de una red y el grafo de interferencia correspondiente, donde dos nodos son adyacentes si las rutas de las demandas asociadas comparten enlaces en la red.*

Una vez construido el grafo de interferencia M, podemos pensar a la asignación de espectro como un coloreo por intervalos sobre M. se puede ver que la siguiente adaptación a la lógica de asignación de espectro del modelo original resuelve el problema en cuestión:

$$
\begin{aligned}
\min \quad & maxS & & & \\
\text{s.a.} \quad & maxS & \geq\ & r_d, & \forall\, d \in D & \quad (4.5)\\
& n_{dd'} + n_{d'd} & \geq\ & M_{d,d'}, & \forall\, d, d' \in D & \quad (4.6)\\
& l_d + v(d) & \leq\ & l_{d'} + \overline{s}(1 - n_{dd'}), & \forall\, d, d^{'} \in D & \quad (4.7)\\
& r_d & =\ & l_d + v(d) - 1, & \forall\, d \in D & \quad (4.8)\\
1 \leq\ & l_d & \leq\ & \overline{s} - v(d) + 1, & \forall\, d \in D & \quad (4.9)\\
& n_{dd} & \in\ & \{0, 1\}, & \forall\, d, d' \in D & \quad (4.10)\\
& l_d, r_d & \in\ & \mathbb{Z}, & \forall\, d \in D & \quad (4.11)
\end{aligned}
$$

Notemos que la función objetivo minimiza el máximo slot usado. En otras palabras, el modelo busca una asignación de slots que los agrupe lo más a la izquierda posible. Esto es altamente deseable en redes reales ya que maximizan el espacio disponible para demandas adicionales que puedan llegar, y de hecho es común ver modelos de programación lineal entera para el RSA que usen esto como función objetivo [24][49][50]. De esta manera, la descomposición propuesta produce soluciones al RSA que no sólo son óptimas en términos

del uso de enlaces de la red, sino también que para dicho ruteo buscan la asignación de slots más condensada posible.

4.5. Producción de cortes

El problema maestro a resolver puede pensarse como un problema de encontrar el ruteo más eficiente que tenga una asignación de slots válida (aún si no busca encontrar tal asignación, ni hace referencia explícita a los slots). Con esta mirada, podemos ver al modelo maestro inicial como una versión muy relajada del problema, que buscaremos fortalecer con planos de corte.

Como mencionamos en la introducción a esta parte del trabajo, si el objetivo final del problema es conseguir una solución que use la menor cantidad posible de enlaces de la red, el ruteo hallado en alguna iteración del algoritmo será el óptimo si (y sólo si) la fase de coloreo encuentra para él una solución factible. Si el problema de coloreo para un ruteo dado no es factible, sin embargo, debemos construir cortes que identifiquen el problema con el ruteo actual y prohíban esas situaciones, y agregarlas al modelo de ruteo. Ante un ruteo problemático que no admite un coloreo factible, hay una variedad de estrategias para intentar robustecer el modelo maestro.

Producción de cortes sin procesamiento del ruteo

i - No-Good

Empezando por la estrategia más ingenua, podemos simplemente pedirle al modelo maestro que dé otro ruteo cualquiera. Dado que la única información nueva que recibe el modelo en este caso es que el ruteo hallado no sirvió, estos cortes suelen llamarse *"No-Good"*. Cortes *No-Good* se caracterizan por ser poco informativos pero baratos de obtener, al no requerir ningún análisis de la solución hallada. Esta estrategia puede ser útil cuando el problema maestro produce de entrada una solución muy cercana a la óptima del problema general y requiere mínimos ajustes para alcanzarla.

Desigualdades *No-Good* para nuestro algoritmo se pueden construir tomando las asignaciones de demandas a arcos que componen el ruteo hallado y pidiendo que no ocurran todas a la vez. Sea A el conjunto de variables y activadas en el ruteo (es decir, el conjunto de asignaciones demandas-a-arco realizadas). Entonces la siguiente restricción nos da lo que queremos:

$$\sum_{y \in A} y < |A|$$

Producción de cortes con procesamiento del ruteo

La principal desventaja del anterior esquema es que, al no brindarle información al problema maestro sobre por qué un ruteo no funcionó como solución, cada iteración del algoritmo poda una sola solución del espacio total.

Para construir cortes más informativos acerca de por qué un ruteo dado no puede ser coloreado, debemos acercarnos lo más posible a la raíz del problema. En otras palabras,

buscamos recortar el conjunto de asignaciones que no pueden ocurrir a la vez. Así, la restricción producida en cada iteración del algoritmo sería lo más general posible, y al aplicar a muchas posibles soluciones, reduciría en gran cantidad el espacio de soluciones.

Todas las estrategias de eliminación de enlaces en este trabajo funcionan de manera similar. Partiendo del ruteo entero como descripción de lo que no puede ocurrir, se elige con algún criterio un enlace para eliminar del ruteo. Se construye a partir de este ruteo reducido el grafo de interferencia de la segunda fase y se intenta colorearlo. De no ser factible el coloreo, sabemos que las eliminaciones realizadas hasta el momento no resuelven el problema; podemos pensar que el conflicto que bloquea el coloreo sigue presente en el ruteo. Este proceso de eliminación de enlaces e intento de coloreo se itera hasta llegar a un ruteo que admite un coloreo factible. Una vez encontrado, se restaura el último enlace eliminado, así destilando la descripción inicial del conflicto a una mucho menos específica.

Para limitar el scope del trabajo, se estudiarán criterios de "caja negra". Es decir, no utilizaremos herramientas de teoría de grafos para sacar conclusiones ad-hoc de cada ruteo en relación a la red. Esto deja una posible avenida de trabajo futuro.

i - Eliminación de asignaciones aleatorias

Como primer acercamiento a la producción de desigualdades informativas, estudiamos la eficiencia del algoritmo cuando el criterio de eliminación simplemente saca asignaciones aleatorias del ruteo. Este enfoque busca un balance entre la cantidad esperada de eliminaciones, y el costo del preprocesamiento realizado en cada iteración.

ii - Enlaces de menor uso

Continuando la búsqueda de desigualdades útiles, implementamos ahora un criterio que realiza un preprocesamiento del ruteo hallado para informar las decisiones de eliminación.

Consideremos un conjunto de asignaciones de demandas a enlaces dentro de una red que no tienen configuración de slots válida. Asumiendo que este conflicto ocurre en secciones concentradas de la red y que hay asignaciones que no influyen en el conflicto, las asignaciones a remover serían estas últimas. Es esperable que haya alguna relación entre el nivel de uso que recibe un enlace y su participación en el conflicto. Buscando validar esta hipótesis, esta implementación hará un procesamiento adicional del ruteo que calcule el flujo que pasa por cada enlace, y luego ordene las asignaciones y_{de} según el uso que las demandas le dan al enlace e.

Eliminando asignaciones en cada paso empezando por aquellas sobre los arcos menos usados, se espera que para el momento en que se encuentre una asignación que, al eliminarse, admita un coloreo, el conjunto de asignaciones restantes sea menor que en la implementación de la sección anterior.

4.6. Resultados computacionales - tiempos de ejecución

4.6.1. Estrategias de producción de cortes

La figura 4.3 muestra, en escala logarítmica, los tiempos de ejecución para una instancia representativa del modelo base DR-AOV y de variantes del modelo con la descomposición basada en Benders, bajo distintas estrategias de producción de cortes.

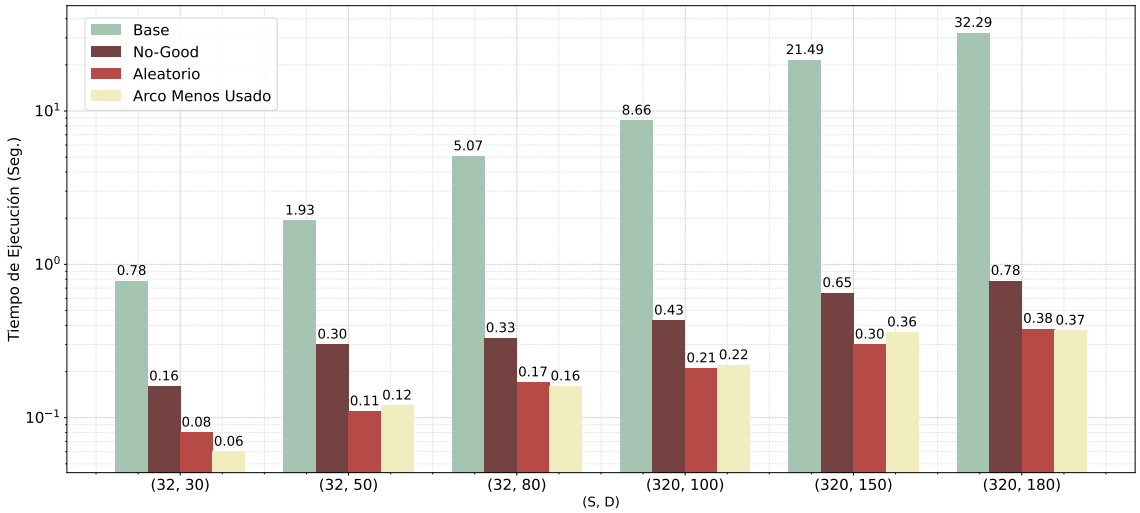

Figura 4.3: *Comparación (en escala logarítmica) de los tiempos de ejecución para el modelo base y variantes del modelo con descomposición. Instancia: 30n-112m-Spain con densidad media.*

Se puede apreciar fácilmente la gran diferencia en los tiempos de ejecución. La enorme ventaja que presenta incluso la más ingenua de las implementaciones pone en evidencia el gran potencial de las descomposiciones propuestas. Adicionalmente, los tiempos dados con la estrategia de eliminación de asignaciones aleatorias muestra el poder de refinar la desigualdad producida en cada iteración del algoritmo, recortando los tiempos de la implementación *naive*, que ya presentaba mejoras en comparación al modelo base.

Por otro lado, la figura muestra el rendimiento decreciente dado por la estrategia del arco menos usado, que realiza un procesamiento más extenso del ruteo producido en cada iteración. Como las dos últimas estrategias presentan tiempos muy similares, está claro que hay un trade-off considerable entre la robustez del refinamiento y las ventajas computacionales que el mismo brinda.

4.6.2. Análisis en función de la densidad

Para realizar un análisis más exhaustivo, estudiamos el rendimiento bajo distintos volúmenes de demanda. Optamos por la estrategia de eliminar las asignaciones sobre arcos de menor uso, por tener mayor relación con las propiedades de las redes y las instancias.

La figura 4.4 compara el rendimiento para tres tipos de densidades: baja, normal, y alta. Lo que se puede ver es que en general, aumentos en la cantidad de demandas implican grandes saltos en tiempos de resolución para el modelo base, y saltos casi despreciables en el modelo con descomposición. Esta brecha culmina en instancias para las cuales el modelo base no puede resolver el problema en el tiempo máximo asignado, mientras que el algoritmo con la descomposición propuesta los resuelve en menos de un segundo.

4.6.3. Densidades realistas

Como último análisis con estas topologías, estudiamos el desempeño de los modelos cuando los volúmenes de las demandas es definido con distribución uniforme en $\{1, 2, 4\}$.

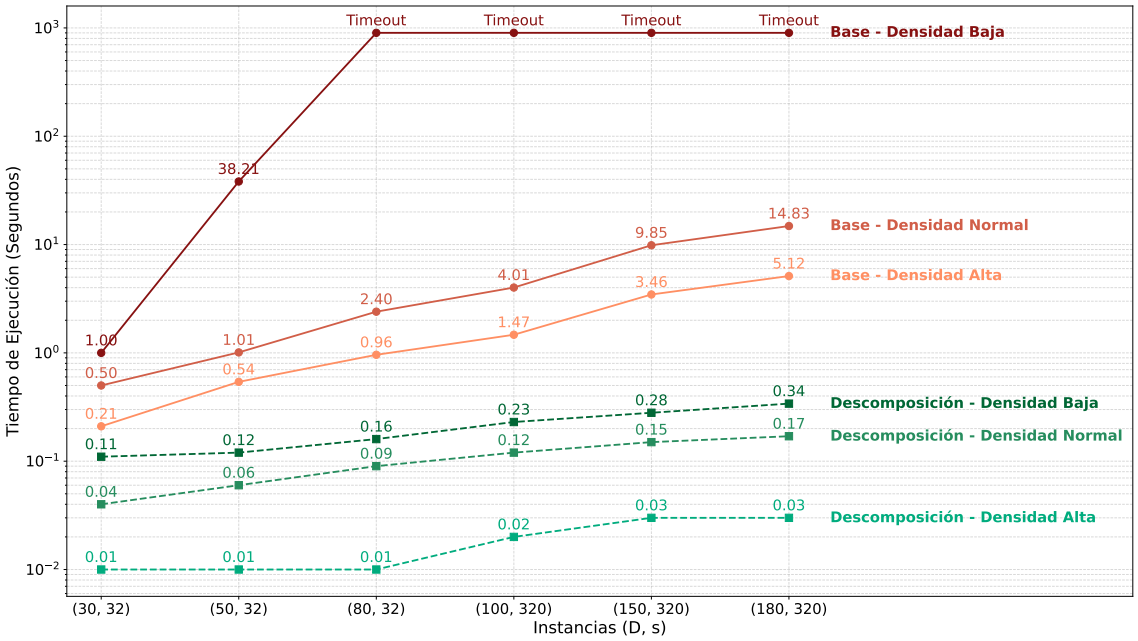

Figura 4.4: *Tiempos de ejecución (en escala logarítmica) del modelo base y algoritmo de descomposición para densidades bajas, normales, y altas. Topología: 24n-86m-UBN24*

Recordamos que esto modela el caso de uso típico donde se tienen slots de 12.5GHz y demandas de 10, 40, y 100Gbps. Para estas densidades consideraremos dos tandas de instancias, con tamaños chicos y grandes de demandas y slots. Es de interés estudiar ambos casos porque como veremos, el modelo con mejor rendimiento cambia en cada caso.

Empezando por la tanda de menor tamaño, la figura 4.5 muestra patrones similares a aquellos de la figura 4.4, con los valores para la descomposición manteniéndose estables mientras que los tiempos para el modelo base crecen considerablemente a medida que aumenta la cantidad de demandas y slots. Esto mantiene el patrón observado en experimentación previa.

Para instancias con más demandas, sin embargo, la situación es por primera vez invertida: si bien el modelo base es capaz de encontrar soluciones óptimas en el tiempo límite cómodamente, al modelo con descomposición le resulta muy difícil obtener una solución óptima, alcanzando rápidamente el límite de tiempo impuesto y provocando timeouts. Esto se puede observar en la figura 4.6. Cabe destacar que estas cifras son para una topología de mucho menor tamaño que el caso anterior, 14n-42m-NSF, y ya aquí se generan timeouts.

Luego de un análisis subsecuente de la descomposición, se identificó que la raíz del problema es la función objetivo del modelo auxiliar; buscar en este modelo un coloreo por intervalos de mínimo ancho resulta ser muy computacionalmente demandante, y empeora significativamente los tiempos de ejecución aún cuando se realiza una sola iteración del algoritmo.

Aquí vale la pena recordar que la búsqueda de un coloreo por intervalos del menor ancho posible no es algo que esté presente en el modelo base, donde el único objetivo es minimizar la cantidad de arcos utilizados, sino que es más bien un refinamiento adicional de la solución que podíamos hacer gracias a que los recortes a los tiempos de ejecución lo

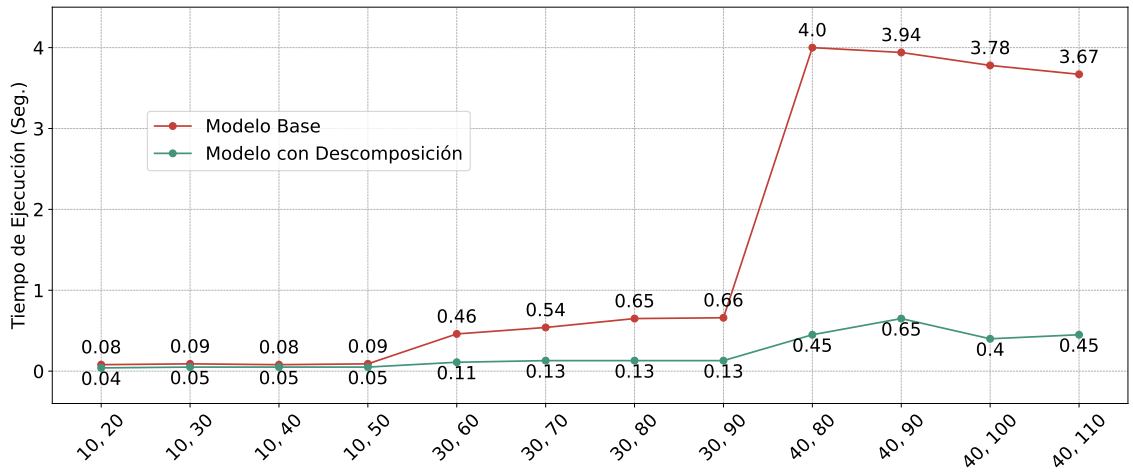

Figura 4.5: *Tiempos de ejecución para instancias chicas densidades con distribución uniforme en* $\{1, 2, 4\}$*. Topología: 43n-176m-Eurolarge.*

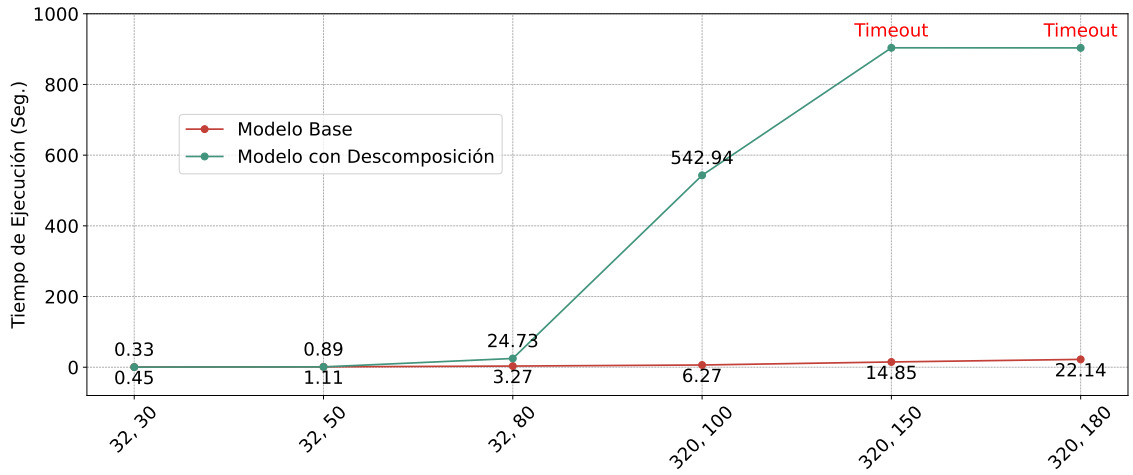

Figura 4.6: *Tiempos de ejecución para instancias grandes y densidades con distribución uniforme en* $\{1, 2, 4\}$*. Topología: 14n-42m-NSF.*

permitían. Decidimos entonces para este caso de prueba en particular prescindir de este refinamiento y buscar un coloreo por intervalos cualquiera. La figura 4.7 muestra que al modificar el modelo auxiliar para devolver el primer coloreo factible hallado, nuevamente la descomposición se vuelve la mejor opción, volviendo a exhibir el patrón que se veía anteriormente en donde los tiempos de ejecución del modelo base crecen mucho más rápidamente que los del modelo con la descomposición aplicada. Aquí nuevamente mostramos cifras para la mayor topología con la que trabajamos, 43n-176m-Eurolarge.

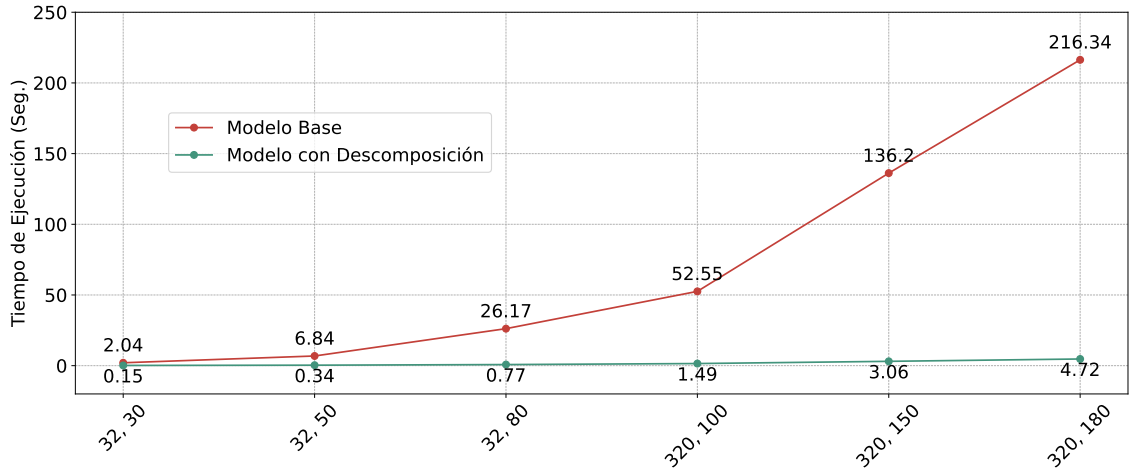

Figura 4.7: *Tiempos de ejecución para instancias grandes y densidades con distribución uniforme en* $\{1, 2, 4\}$ *(variante con primer coloreo factible). Topología: 43n-176m-Eurolarge.*

En definitiva, vemos que la descomposición permite en la gran mayoría de los casos obtener soluciones óptimas en mucho menor tiempo que el modelo base. Más aún, como el modelo auxiliar busca minimizar el span de slots usado, las soluciones halladas por este modelo optimizan una cualidad que es altamente deseable en las EONs.

4.6.4. Instancias sintéticas

El procedimiento basado en la descomposición combinatoria de Benders desarrollado en este trabajo trivializa muchas de las instancias que anteriormente presentaban problemas, ya sea en tiempos de ejecución imprácticos o en requisitos de memoria inalcanzables. En particular, reduce enormemente los tiempos de ejecución para las mayores topologías usadas, que incidentalmente son las mayores topologías encontradas en la literatura.

Con el fin de expandir la frontera de instancias para las cuales RSA puede ser resuelto de manera exacta con modelos PLE en tiempos prácticos, construimos topologías sintéticas que si bien no corresponden a EONs reales, capturan sus principales propiedades. Generando instancias para estas topologías, podemos obtener un buen indicio del desempeño que tendrían estos modelos al enfrentarse con casos de uso que exceden la capacidad de los modelos actuales.

Para construir instancias sintéticas que preserven las propiedades reales de las redes existentes, analizamos tres métricas que surgen del diseño de redes y que consideramos relevantes para el problema: el grado promedio y centralidad de cada nodo, y la distancia promedio entre cada par de nodos. Al computar estas métricas para las redes existentes y

Topología	Grado Promedio	Distancia Promedio	Máxima Centralidad
6n-9m-n6s9	3.00	1.47	0.50
10n-44m-SmallNet	4.40	1.58	0.41
11n-52m-Pan-European-COST239	4.73	1.56	0.38
14n-42m-NSF	3.00	2.14	0.34
14n-46m-Generic-Deutsche-Telekom-DT	3.29	2.34	0.47
15n-46m-NSF	3.07	2.24	0.37
16n-46m-EURO	2.88	2.64	0.40
20n-62m-ARPANet	3.10	2.81	0.36
20n-78m-EON20	3.90	2.36	0.37
21n-70m-SpanishTelefonica	3.33	2.76	0.31
21n-72m-Italian	3.43	2.92	0.39
21n-78m-UKNet	3.71	2.50	0.41
22n-70m-British-telecom	3.18	3.00	0.37
24n-86m-UBN24	3.58	3.03	0.36
28n-68m-EON	2.43	4.00	0.33
28n-82m-EURO28	2.93	3.56	0.33
30n-112m-Spain	3.73	3.31	0.28
43n-176m-EuroLarge	4.09	3.64	0.30
46n-186m-Synthetic	4.17	3.35	0.30
53n-214m-Synthetic	4.04	3.46	0.33
64n-266m-Synthetic	4.16	3.48	0.28

Cuadro 4.1: *Topologías existentes y nuevas, junto a sus valores para tres propiedades.*

observar su cambio (o ausencia del mismo) a medida que las topologías crecen, obtuvimos criterios para determinar si una topología sintética representa con fidelidad una posible red real del mismo tamaño.

En este trabajo proponemos tres topologías de prueba, cuyos tamaños son:

- 46 nodos y 186 arcos
- 53 nodos y 214 arcos
- 64 nodos y 266 arcos

La tabla 4.1 muestra los valores de dichas métricas para las redes reales con las que trabajamos y para las tres topologías sintéticas que proponemos.

Consideramos que las tres redes propuestas definen redes plausibles; los grados promedios caen en el rango de las redes reales, y la distancia promedio y máxima centralidad continúan la tendencia de cambio que se observa a medida que crecen las redes.

Para evaluar el modelo base y el modelo con la descomposición basada en Benders para estas topologías, generamos una suite de instancias que contempla varios rangos de densidad.

Densidad baja

Consideramos primero las instancias con densidades bajas. Cabe destacar que en experimentos previos, notamos que aquí residía el punto límite después del cual el modelo base no era capaz de resolver las instancias en el tiempo permitido. La tabla 4.8

$\overline{s}$	*32*			*320*		
$\|D\|$	*30*	*50*	*80*	*100*	*150*	*180*
Modelo Base						
46n-186m-Synthetic	2.20	6.82	23.83	334.54	Timeout	Timeout
53n-214m-Synthetic	2.48	6.89	321.61	Timeout	Timeout	Timeout
64n-266m-Synthetic	3.20	9.17	35.99	343.75	Timeout	Timeout
Modelo con Descomposición						
46n-186m-Synthetic	0.19	0.43	3.09	8.32	249.81	301.89
53n-214m-Synthetic	0.22	0.65	4.73	7.32	787.43	761.45
64n-266m-Synthetic	0.25	0.54	2.12	8.92	69.10	677.38

Figura 4.8: *Comparación del Modelo Base y el Modelo con Descomposición sobre instancias sintéticas con densidad baja.*

A primera vista, la mayor diferencia entre ambas implementaciones es que aquella con la descomposición de Benders fue capaz de resolver todas las instancias en el tiempo designado, mientras que el modelo base no logró resolver las de mayor tamaño. En las instancias que el modelo base sí pudo resolver, pasar a la implementación optimizada produce un recorte aproximado del 97 % en los tiempos de ejecución.

Densidad normal

La tabla 4.9 muestra los resultados de estos experimentos para densidades normales.

$\overline{s}$	*32*			*320*		
$\|D\|$	*30*	*50*	*80*	*100*	*150*	*180*
Modelo Base						
46n-186m-Synthetic	1.26	301.81	3.31	1.02	2.47	3.64
53n-214m-Synthetic	1.33	25.01	9.11	13.15	36.76	52.14
64n-266m-Synthetic	1.53	16.72	12.32	17.43	43.84	72.45
Modelo con Descomposición						
46n-186m-Synthetic	0.26	0.32	0.10	0.11	0.02	0.03
53n-214m-Synthetic	0.22	0.51	0.39	0.39	0.54	0.69
64n-266m-Synthetic	0.25	0.69	0.49	0.52	0.71	0.89

Figura 4.9: *Comparación del Modelo Base y el Modelo con Descomposición sobre instancias sintéticas con densidad media.*

Lo que se puede ver al pasar a esta densidad es, primero, que el modelo base es ahora capaz de resolver todas las instancias dentro del tiempo límite. Sin embargo, sigue presentando dificultades considerables para resolver ciertas instancias.

Por otro lado, la descomposición reduce drásticamente los tiempos de ejecución al pasar a esta densidad, manteniéndose en este caso por debajo de un segundo en todas las instancias. Se observó que con demandas de esta densidad, el primer ruteo propuesto por

el modelo maestro era casi siempre factible. Evidentemente estas densidades hacen que un ruteo que no sature ningún arco individualmente admita con alta probabilidad un coloreo factible en la segunda fase del algoritmo. Con esta densidad, el recorte promedio de tiempos de ejecución al implementar la descomposición ronda el 95 %.

Densidad Alta

El último análisis hecho en esta sección, en este caso para densidades altas, es presentado en la tabla 4.10.

$\overline{s}$	*32*			*320*		
$\|D\|$	*30*	*50*	*80*	*100*	*150*	*180*
Modelo Base						
46n-186m-Synthetic	0.26	1.02	0.65	1.06	2.59	3.72
53n-214m-Synthetic	0.40	1.12	2.89	4.23	10.38	14.83
64n-266m-Synthetic	0.49	1.56	3.65	5.49	13.27	19.44
Modelo con Descomposición						
46n-186m-Synthetic	0.08	0.18	0.01	0.12	0.02	0.03
53n-214m-Synthetic	0.13	0.20	0.30	0.35	0.54	0.70
64n-266m-Synthetic	0.26	0.22	0.35	0.42	0.70	0.86

Figura 4.10: *Comparación del Modelo Base y el Modelo con Descomposición sobre instancias sintéticas con densidad alta.*

Para estas densidades, los tiempos de ejecución de la implementación optimizada presentan leves diferencias respecto a las densidades anteriores; esto es esperable, dado que en ambos casos resulta altamente probable que el primer ruteo hallado admita un coloreo factible. Por otro lado, los tiempos para el modelo base nuevamente muestran una reducción considerable, ya que es más fácil detectar instancias infactibles a medida que crecen los volúmenes de cada demanda. De esta manera, la brecha en tiempos de ejecución se achica y se sitúa ahora en un valor promedio de 87 %.

4.7. Resultados computacionales - consumo de memoria

Además de las ventajas ofrecidas en términos del tiempo de resolución de los problemas, otra ventaja importante ofrecida por las descomposiciones es el uso más eficientes de recursos computacionales. Los problemas resultantes de una descomposición – cada uno considerablemente menos complejo que el problema completo – son resueltos uno a la vez, lo cual reduce los recursos demandados en un momento dado de la computadora que los esté resolviendo. En esta sección estudiamos la memoria utilizada por el modelo base y el algoritmo de descomposición, para evaluar la magnitud de los ahorros de memoria obtenidos por esta última.

Comenzamos estudiando el consumo de memoria para la topología 43n-176m-Eurolarge. Consideramos que es de interés por ser la mayor de las topologías de la literatura basadas en redes reales. La figura 4.11 muestra los resultados obtenidos para este caso.

Se puede ver fácilmente que la descomposición genera grandes ahorros de memoria en todas las instancias. Por un lado, el modelo base impone requerimientos de memoria

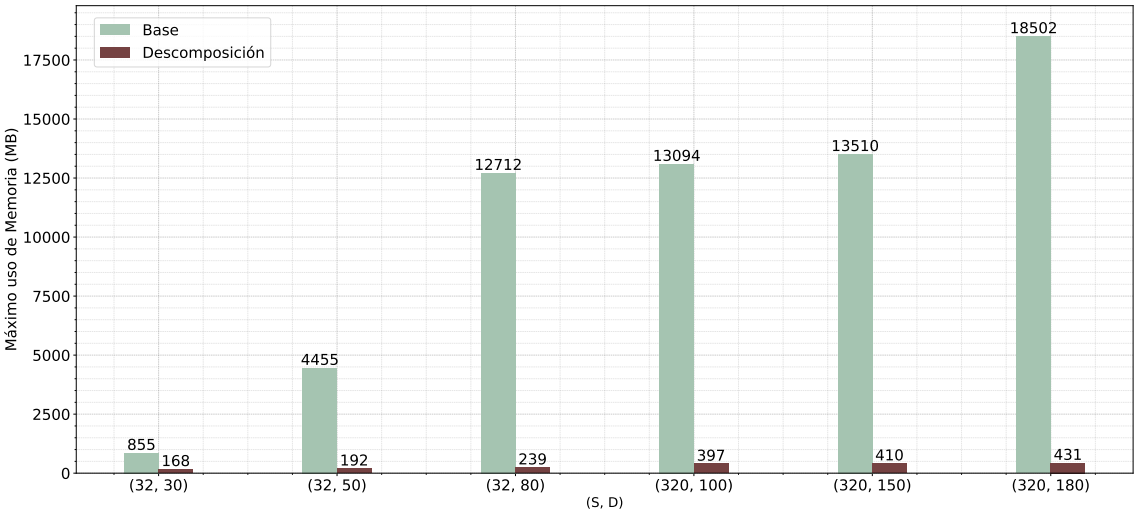

Figura 4.11: *Consumo de memoria del modelo AOV base y del algoritmo de descomposición para distintas instancias. Topología: 43n-176m-Eurolarge. Densidades realistas.*

que crecen muy rápidamente a medida que aumenta la cantidad de demandas, utilizando 855MB en la más chica de las instancias y alcanzando en el peor de los casos un pico de uso de 18.5GB. La descomposición, en cambio, exhibe incrementos de utilización de memoria mucho menores, requiriendo 168MB en la instancia más chica y 431MB en la más grande.

Continuamos el análisis con la más grande de las tres topologías sintéticas propuestas en el trabajo: 64n-266m-Synthetic. Recordamos que con esta topología, la versión del algoritmo de descomposición que buscaba minimizar el ancho del coloreo por intervalos exhibió tiempos de ejecución significativamente peores a los del modelo base. En consecuencia, decidimos analizar el uso de memoria para ambas variantes del algoritmo. Los resultados de este análisis se encuentran en la figura 4.12.

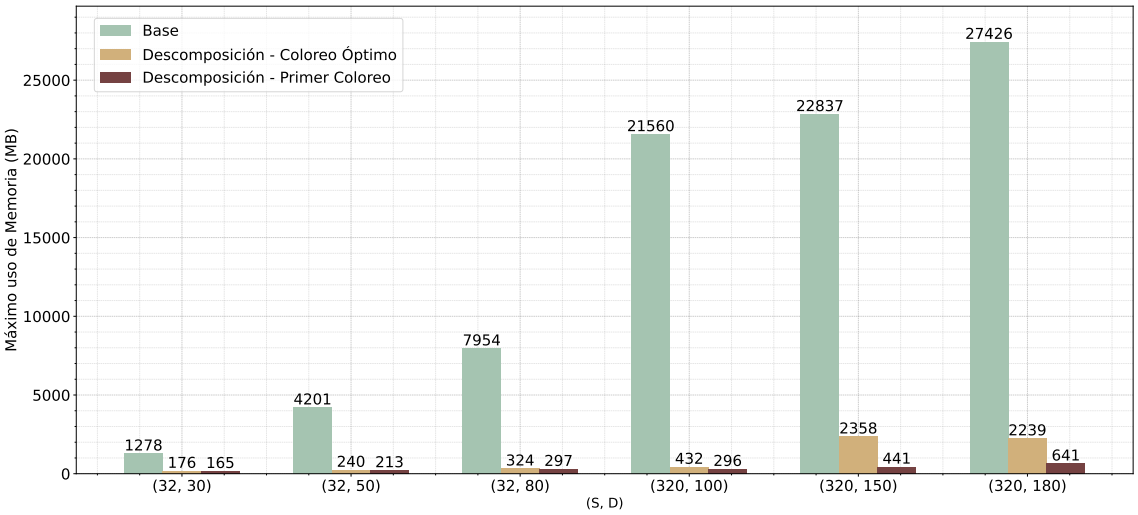

Figura 4.12: *Consumo de memoria del modelo AOV base y del algoritmo de descomposición para distintas instancias. Topología: 64n-266m-Synthetic. Densidades realistas.*

Los resultados de este experimento muestran no solamente que las grandes ventajas en

complejidad espacial se mantienen para topologías de este tamaño, sino que en este caso no se observa una diferencia tan grande como en la sección anterior al modificar la función objetivo del modelo auxiliar. Es decir, lo que se ve es que la necesidad de prescindir de coloreos óptimos es causada principalmente por los tiempos de ejecución, y no por el uso de memoria.

En general, consideramos que estos resultados son muy positivos. Disminuir drásticamente los requisitos de capacidad computacional implica la posibilidad de resolver el RSA en sistemas mucho más económicos, aún en las instancias más difíciles del mismo.

Capítulo 5

Descomposiciones para S-RSA con DPP y SBPP

5.1. Descomposición para DPP

Dado el éxito de la descomposición combinatoria de Benders propuesta para el modelo que resuelve el problema RSA base, decidimos adicionalmente estudiar su desempeño en los modelos que resuelven RSA con Path Protection. Empezaremos con el modelo que provee protección de caminos dedicada y luego estudiaremos cómo la descomposición se comporta con la protección de caminos compartida.

5.1.1. Modelo maestro

De manera similar a la descomposición presentada anteriormente, el modelo maestro no es más que una versión reducida de DPP que no considera la asignación de slots, encargándose únicamente de hallar caminos para cada demanda. Aquí también agregamos una restricción que les da un *warm-start* a los caminos que devuelve.

$$\min \quad \sum_{d\in D}\sum_{e\in E} (ym_{de} + yb_{de})$$

$$\text{s.a.} \sum_{e\in\delta^+(v)} ym_{de} - \sum_{e\in\delta^-(v)} ym_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{en caso contrario} \end{cases}, \qquad \forall d \in D, v \in V \tag{5.1}$$

$$\sum_{e\in\delta^+(v)} yb_{de} - \sum_{e\in\delta^-(v)} yb_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{en caso contrario} \end{cases}, \qquad \forall d \in D, v \in V \tag{5.2}$$

$$ym_{de} + yb_{de} \leq 1, \qquad \forall d \in D, e \in E \tag{5.3}$$

$$\sum_{d\in D} yb_{de} \leq 1, \qquad \forall e \in E \tag{5.4}$$

$$\sum_{d\in D} v(d)\ (ym_{de} + yb_{de}) \leq S, \qquad \forall e \in E \tag{5.5}$$

5.1.2. Modelo auxiliar

Para el modelo auxiliar, proponemos nuevamente un grafo de interferencia sobre el cual realizaremos un coloreo por intervalos. Como antes, tendremos nodos para las demandas y modelaremos el cruce entre dos caminos agregando una arista entre sus respectivas demandas en el grafo de interferencia. Algo a notar, sin embargo, es que en estos modelos cada demanda cuenta con dos caminos: uno titular y uno de respaldo. Al momento de realizar la asignación de slots, es necesario contar con la información sobre ambos de estos caminos para cada demanda. Así, nuestro grafo de interferencia tendrá para cada demanda un nodo correspondiente a su camino titular y uno correspondiente a su camino de respaldo:

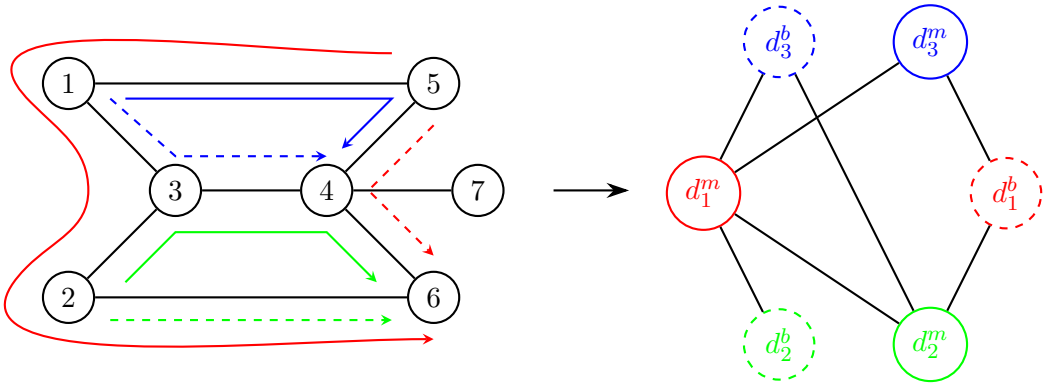

Figura 5.1: *El ruteo de tres demandas, cada una con camino titular y de respaldo (respectivamente representados con una curva lisa y punteada), y el grafo de interferencia asociado. En este caso se definen dos nodos por demanda, uno por cada camino.*

Construida la matriz M de interferencia descripta arriba, definimos el modelo:

$$
\begin{array}{lrcll}
\min & maxS & & & \\
\text{s.a.} & maxS & \geq & rm_d, & \forall d \in D \quad (5.6)\\
 & maxS & \geq & rb_d, & \forall d \in D \quad (5.7)\\
n^{mm}_{dd'} + n^{mm}_{d'd} & & \geq & M_{d_m d'_m}, & \forall d, d' \in D \quad (5.8)\\
n^{mb}_{dd'} + n^{bm}_{d'd} & & \geq & M_{d_m d'_b}, & \forall d, d' \in D \quad (5.9)\\
lm_d + v(d) & & \leq & lm_{d'} + \overline{s}(1 - n^{mm}_{dd'}), & \forall d, d' \in D \quad (5.10)\\
lm_d + v(d) & & \leq & lb_{d'} + \overline{s}(1 - n^{mb}_{dd'}), & \forall d, d' \in D \quad (5.11)\\
lb_d + v(d) & & \leq & lm_{d'} + \overline{s}(1 - n^{bm}_{dd'}), & \forall d, d' \in D \quad (5.12)\\
rm_d & & = & lm_d + v(d) - 1, & \forall d \in D \quad (5.13)\\
rb_d & & = & lb_d + v(d) - 1, & \forall d \in D \quad (5.14)\\
lm_d & & \geq & 1, & \forall d \in D \quad (5.15)\\
lb_d & & \geq & 1, & \forall d \in D \quad (5.16)\\
lm_d & & \leq & \overline{s} - v(d) + 1, & \forall d \in D \quad (5.17)\\
lb_d & & \leq & \overline{s} - v(d) + 1, & \forall d \in D \quad (5.18)
\end{array}
$$

5.1.3. Resultados computacionales- Tiempos de ejecución

En esta sección consideraremos las mismas estrategias de producción de cortes que comentamos en el capítulo anterior. La primera de estas, y también la más ingenua, es la estrategia *No-Good*, en la cual simplemente se le pide al modelo maestro un ruteo distinto a los que ya produjo. Las otras dos buscan refinar el corte producido eliminando asignaciones de demandas a arcos iterativamente, frenando al detectar que el ruteo se volvió factible y restaurando la última asignación eliminada. Estas son las estrategias *Eliminación Aleatoria* y *Eliminación por arco menos usado*, y como el nombre sugiere, buscan respectivamente eliminar en cada iteración una asignación aleatoria y una asignación realizada sobre el arco menos usado por el ruteo. La tabla 5.2 muestra los resultados con la descomposición de DPP, comparando las estrategias entre sí y además contra la implementación base.

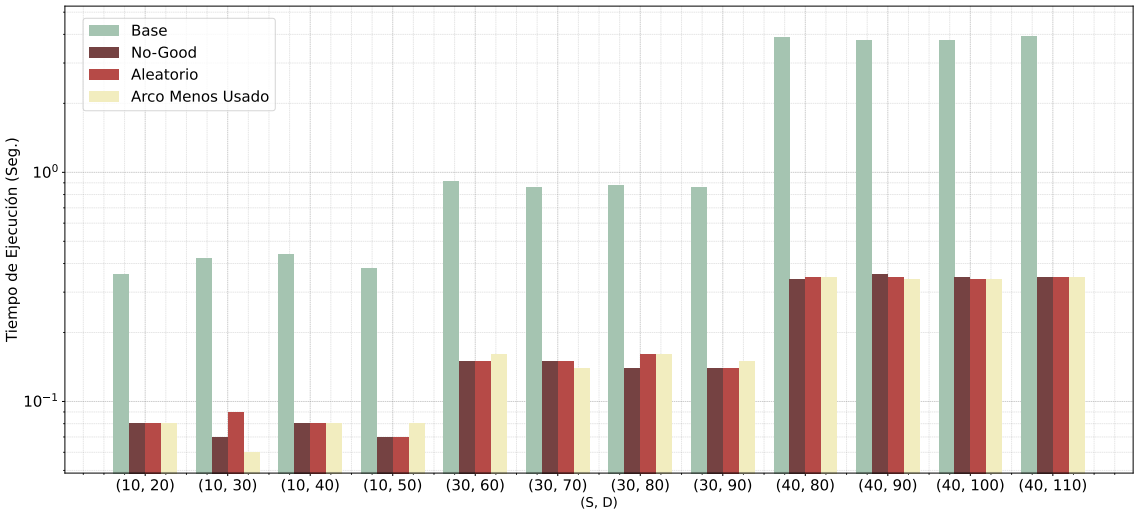

Figura 5.2: *Tiempos de ejecución en escala logarítmica para cada una de las estrategias de producción de cortes. Se incluye también el desempeño del modelo original. Modelo: DPP*

Nuevamente, el recorte dado por la descomposición es claro; la reducción en tiempos alcanza el 96 %. La descomposición claramente es también la opción ganadora en este caso, reduciendo el tiempo requerido para resolver las instancias y además logrando resolver instancias que antes alcanzaban el límite de tiempo impuesto.

Sin embargo, en este caso – a diferencia del modelo AOV base – no hay una diferencia notable entre las distintas estrategias. En un análisis posterior realizado sobre las ejecuciones, notamos que en la gran mayoría de las instancias factibles, el primer ruteo hallado admitía un coloreo válido y luego conducía a una solución óptima al problema completo. Esta falta de necesidad de producir cortes produce resultados muy similares para todas las estrategias.

5.1.4. Resultados computacionales - consumo de memoria

Al igual que en el capítulo anterior, estudiaremos también el consumo de memoria del modelo DPP base y del algoritmo basado en su descomposición. La esperanza es que los resultados del capítulo anterior se mantengan y presenten aún más ventajas de la descomposición por sobre el modelo base.

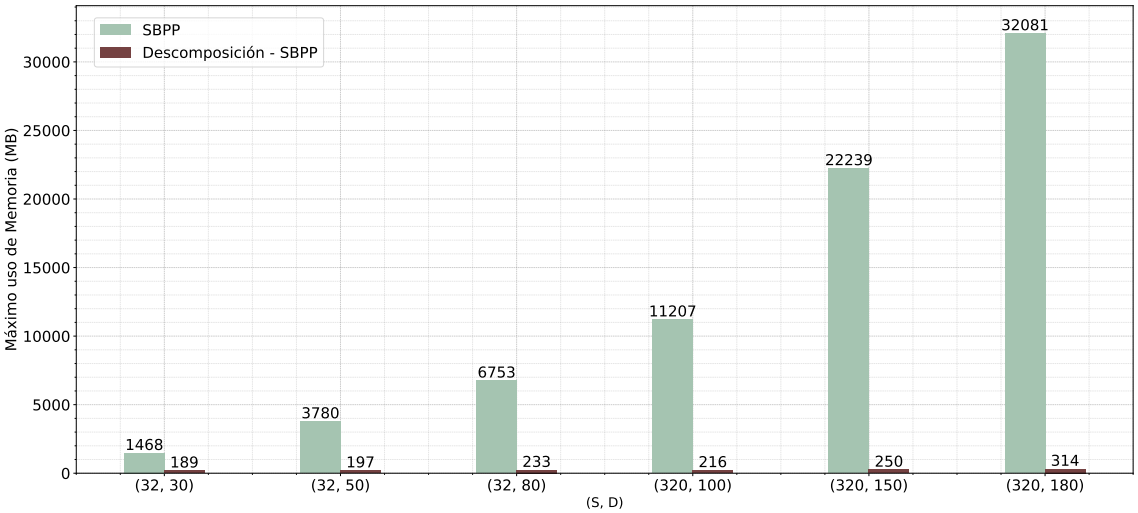

Figura 5.3: *Consumo de memoria del modelo DPP base y del algoritmo de descomposición para distintas instancias. Topología: 43n-176m-Eurolarge. Densidades realistas.*

Los resultados de estas pruebas se pueden ver en la figura 5.3. Se puede ver ahí que efectivamente la descomposición sigue recortando enormemente los requisitos de memoria necesarios para resolver las instancias, exhibiendo por ejemplo un recorte del 99 % para la más grande de estas.

5.2. Descomposición para SBPP

Esta sección contiene la última descomposición propuesta en este trabajo. En este caso aplicaremos las mismas ideas presentadas en este capítulo y el anterior al modelo SBPP, con la esperanza de nuevamente obtener un modelo mucho más eficiente y poder resolver, en este caso, S-RSA con protección vía caminos de respaldo compartidos mucho más rápidamente.

5.2.1. Modelo maestro

El procedimiento para descomponer SBPP en dos modelos separados es muy similar al caso anterior. El modelo maestro será, inicialmente, la parte de SBPP relacionada al ruteo, sin consideración por la configuración de slots. Aquí agregaremos nuevamente una restricción adicional para que el flujo que pasa por todos los arcos sea menor o igual a su capacidad, para que los ruteos hallados partan desde un lugar razonablemente bueno:

$$\min \quad \sum_{d\in D}\sum_{e\in E} (ym_{de} + yb_{de})$$

$$\text{s.a.} \sum_{e\in\delta^+(v)} ym_{de} - \sum_{e\in\delta^-(v)} ym_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{en caso contrario} \end{cases}, \quad \forall d \in D, v \in V \tag{5.19}$$

$$\sum_{e\in\delta^+(v)} yb_{de} - \sum_{e\in\delta^-(v)} yb_{de} = \begin{cases} 1 & \text{if } v = s(d) \\ -1 & \text{if } v = t(d) \\ 0 & \text{en caso contrario} \end{cases}, \quad \forall d \in D, v \in V \tag{5.20}$$

$$ym_{de} + yb_{de} \leq 1, \qquad \forall d \in D, e \in E \tag{5.21}$$

$$p_{dd'} \geq yb_{de} + yb_{d'e} - 1, \qquad \forall d, d^{'} \in D, e \in E \tag{5.22}$$

$$2 - \sum_{e\in\delta^-(v)} ym_{de} - \sum_{e\in\delta^-(v)} ym_{d'e} \geq p_{dd'}, \qquad \forall d, d' \in D, v \in V \tag{5.23}$$

$$\sum_{d\in D} v(d)\ (ym_{de} + yb_{de}) \leq S, \qquad \forall e \in E \tag{5.24}$$

Como siempre, en el correr del algoritmo este modelo podrá verse robustecido con cortes que informen por qué el último ruteo hallado no admitió un coloreo factible.

5.2.2. Modelo auxiliar

Al igual que para las otras descomposiciones, el modelo auxiliar se encargará de definir una configuración de slots para el ruteo hallado por el modelo maestro. Esto, como siempre, lo hará a partir del grafo de interferencia asociado al ruteo hallado:

$$
\begin{aligned}
\min \quad & maxS & & & \\
\text{s.a.} \quad & maxS \geq rm_d, & & \forall d \in D & (5.25)\\
& maxS \geq rb_d, & & \forall d \in D & (5.26)\\
n^{mm}_{dd'} + n^{mm}_{d'd} &\geq M_{d_m d'_m}, & & \forall d, d' \in D, e \in E & (5.27)\\
n^{mb}_{dd'} + n^{bm}_{d'd} &\geq M_{d_m d'_b}, & & \forall d, d' \in D, e \in E & (5.28)\\
n^{bb}_{dd'} + n^{bb}_{d'd} &\geq M_{d_b d'_b}, & & \forall d, d' \in D, e \in E & (5.29)\\
lm_d + v(d) &\leq lm_{d'} + \overline{s}(1 - n^{mm}_{dd'}), & & \forall d, d' \in D & (5.30)\\
lm_d + v(d) &\leq lb_{d'} + \overline{s}(1 - n^{mb}_{dd'}), & & \forall d, d' \in D & (5.31)\\
lb_d + v(d) &\leq lm_{d'} + \overline{s}(1 - n^{bm}_{dd'}), & & \forall d, d' \in D & (5.32)\\
lb_d + v(d) &\leq lb_{d'} + \overline{s}(1 - n^{bb}_{dd'}), & & \forall d, d' \in D & (5.33)\\
rm_d &= lm_d + v(d) - 1, & & \forall d \in D & (5.34)\\
rb_d &= lb_d + v(d) - 1, & & \forall d \in D & (5.35)\\
lm_d &\geq 1, & & \forall d \in D & (5.36)\\
lb_d &\geq 1, & & \forall d \in D & (5.37)\\
lm_d &\leq \overline{s} - v(d) + 1, & & \forall d \in D & (5.38)\\
lb_d &\leq \overline{s} - v(d) + 1, & & \forall d \in D & (5.39)
\end{aligned}
$$

5.2.3. Resultados computacionales - Tiempos de ejecución

El último análisis empírico que realizamos en este trabajo estudiará el desempeño de la descomposición descripta al modelo SBPP. Nuevamente, las instancias usadas en esta sección tendrán densidades reales ($d \in [1, 2, 4]$). Esto se debe a que densidades más grandes reducen fuertemente la esperanza de factibilidad en la fase de ruteo, lo cual hace obsoleta la comparación entre estrategias de producciones de cortes.

Como siempre, empezamos comparando el rendimiento del modelo base contra distintas variantes de la descomposición. En estas variantes, modificamos la estrategia a través de la cual generamos cortes en cada iteración del algoritmo. La figura 5.4 muestra una situación muy similar a la sección anterior, que descomponía DPP: por un lado, la ganancia obtenida por la descomposición, en cualquier variante de la misma, es obvia. Sin embargo, no hay una diferencia significativa entre las distintas estrategias para producir cortes.

Analizando los resultados de las ejecuciones, vemos que esto se debe a que nuevamente los modelos lograban obtener en la enorme mayoría de los casos una solución factible en una sola iteración del algoritmo, y luego no había necesidad de producir cortes en primer lugar.

Lo que se ve por otro lado es que las ventajas ofrecidas por la descomposición en este caso no se comparan a los resultados obtenidos anteriormente. En el peor de los casos, para las instancias de mayor cantidad de demandas, el algoritmo no es capaz de obtener soluciones en el tiempo límite bajo ninguna estrategia. Un análisis de esta situación reveló que el problema no ocurre al llegar a la etapa de coloreo, si no que ya el ruteo es computacionalmente difícil de resolver. En muchos casos, de hecho, se alcanzaba un timeout aún cuando no existía un ruteo factible y luego no se avanzaba a la etapa de

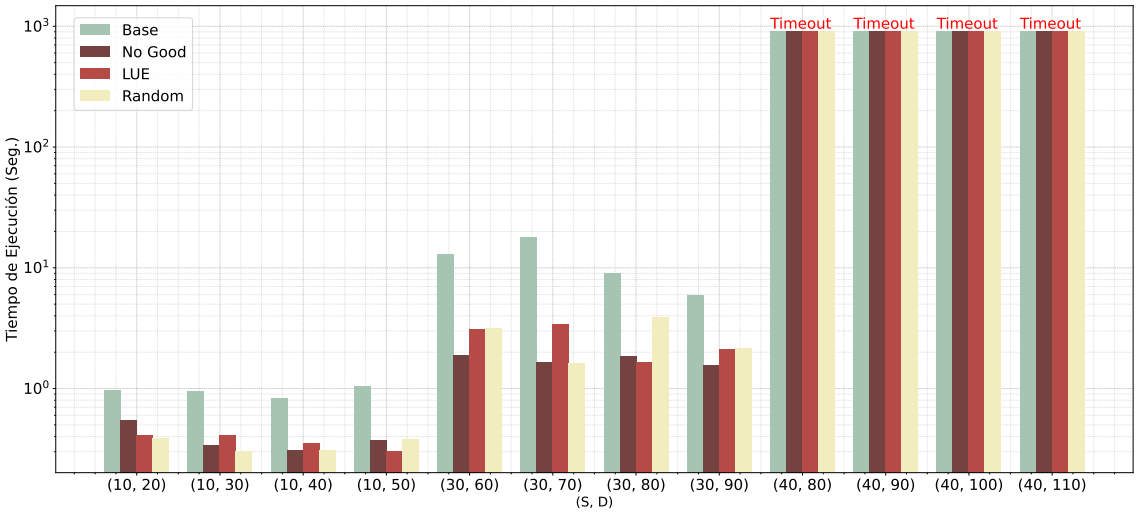

Figura 5.4: *Tiempos de ejecución en escala logarítmica para cada una de las estrategias de producción de cortes. Se incluye también el desempeño del modelo original. Modelo: SBPP*

coloreo. Dado que esto ocurre principalmente en instancias grandes, una optimización posible es incorporar las heurísticas primales diseñadas en el capítulo 3 de este trabajo.

5.2.4. Resultados computacionales - consumo de memoria

Cerramos la tanda de experimentación para este modelo estudiando el consumo de memoria de SBPP y de su descomposición. Esto nos dará una visión más completa de cómo estos se comparan.

Los resultados de dicho análisis se encuentran en la figura 5.5. Similarmente a las cifras para los tiempos de ejecución, este gráfico muestra las mejoras menos significativas de todos los modelos a los cuales aplicamos nuestro esquema de descomposición. No obstante, la mejora sigue siendo fácilmente apreciable, permitiendo resolver las instancias más grandes con la mitad de recursos de memoria que el modelo SBPP base. Cabe destacar que el pico de memoria en la descomposición se debía a la fase de ruteo, que por su complejidad resultaba más demandante en recursos que la fase de coloreo.

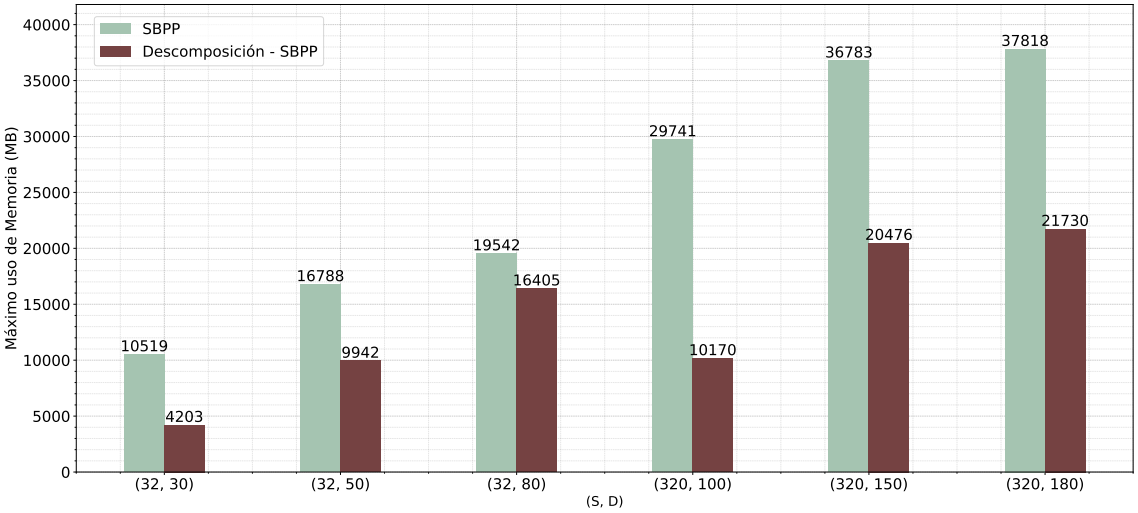

Figura 5.5: *Consumo de memoria del modelo DPP base y del algoritmo de descomposición para distintas instancias. Topología: 43n-176m-Eurolarge. Densidades realistas.*

Capítulo 6

Conclusiones y trabajo futuro

El problema de Ruteo y Asignación de Espectro surge a partir de una de las soluciones más prometedoras para lidiar con las crecientes demandas tráfico en redes de fibra óptica: la tecnología flexgrid. En este trabajo atacamos una generalización de este problema que considera la tolerancia a fallas a través de – según cada variante del problema – distintas estrategias de protección de caminos. A continuación detallamos los principales resultados de este trabajo y deliberamos sobre posibles avenidas de trabajo futuro.

6.1. Conclusiones

Los modelos En este trabajo propusimos dos modelos de programación lineal entera para el Survivable Routing and Spectrum Assignment problem: uno que dota a las soluciones con protección de camino dedicada, de manera que cada demanda es asignada un camino de respaldo exclusivo que ninguna otra demanda puede usar como respaldo, y uno que consigue protección de caminos compartida, donde cada demanda es asignada un camino de respaldo y estos pueden compartir recursos siempre y cuando los caminos titulares de las respectivas demandas sean disjuntos. Basándonos en modelos para RSA propuestos en [23], obtuvimos modelos con desempeño satisfactorio para instancias de tamaño pequeño y mediano. Sin embargo, considerábamos que en las instancias más difíciles había lugar para mejoras (en particular para SBPP), y buscamos en las secciones siguientes implementar distintas optimizaciones.

Las heurísticas Al estudiar el desempeño de los modelos propuestos, se encontraron instancias en las cuales los modelos hallaban rápidamente una buena cota dual para las soluciones, pero requerían mucho tiempo adicional para encontrar una primera solución factible al problema primal. Más aún, notamos que en muchos de estos casos dicha solución factible era rápidamente optimizada una vez encontrada. Para solventar estas dificultades, propusimos una heurística que busca resolver una versión reducida del problema, en la cual ciertas demandas ya tienen predefinido un conjunto limitado de posibles caminos que pueden tomar. Aquí, estas soluciones iniciales – idealmente obtenidas más rápidamente que por los modelos originales – eran luego usadas como *warm-start* para resolver el problema completo.

En este trabajo logramos construir modelos reducidos que en muchos casos reducen significativamente el tiempo total de obtención de soluciones óptimas. Un análisis de los resultados observó mejoras que rondan el 70 % en los tiempos totales de resolución de las

instancias más grandes. Adicionalmente, algunas de las configuraciones halladas también eran capaces de también reducir los tiempos de ejecución de las instancias chicas, en estos casos con mejoras típicamente cercanas al 30 %.

La descomposición Alejándonos del diseño de modelos, y haciendo un cambio de enfoque hacia maneras más sofisticadas de resolver los modelos ya propuestos, diseñamos en este trabajo procedimientos basados en la descomposición combinatoria de Benders tanto para RSA como el S-RSA. Esta descomposición busca resolver por separado el problema de ruteo (R) y el de asignación de espectro (SA), e implementar un esquema iterativo en donde la solución de un problema informe la siguiente resolución del otro problema. Esto se implementó definiendo un modelo de programación lineal entera para cada problema, cada uno basado en los modelos originalmente propuestos para el problema completo. El modelo maestro, encargado de encontrar un ruteo que admitía una asignación de slots factible, le entrega el ruteo hallado al modelo auxiliar, encargado de la configuración de los slots. De hallar el modelo auxiliar una solución factible, sabemos que el ruteo hallado es óptimo y podemos entregar la solución completa. De no haber una configuración de slots factible, el algoritmo debe identificar, con la mayor especificidad posible, el conflicto en el ruteo que prohíbe la factibilidad, y agregar una restricción adicional al modelo maestro que impida dicho conflicto. El modelo maestro produce con esta nueva información un nuevo ruteo y así el ciclo comienza nuevamente.

Estas descomposiciones brindaron por amplia diferencia los mejores resultados de este trabajo, produciendo recortes de más del 95 % en los tiempos de resolución tanto para RSA como S-RSA. Además notamos un gran ahorro en consumo de memoria en los 3 modelos analizados. Esto nos permitió resolver de manera exacta y en un plazo de tiempo práctico instancias más grandes que aquellas encontradas en la literatura. Aquí destacamos el rendimiento del esquema de descomposición al ser aplicado a AOV y DPP. Al aplicarlo a SBPP, las ventajas observadas no fueron tan significativas como en los otros dos modelos. De cualquier manera, la descomposición definitivamente resultó ser la opción ganadora para los tres modelos.

6.2. Trabajo futuro

6.2.1. Modelos con protección de caminos

Si bien a lo largo del trabajo logramos optimizar significativamente el desempeño de los modelos, no estudiamos una avenida común de trabajo: la introducción de desigualdades válidas. Sospechamos que estas podrían mejorar los modelos porque, por un lado, introducir desigualdades válidas a los modelos base dio muy buenos resultados. Adicionalmente, los resultados computacionales hallados mostraron dificultad ocasional en encontrar buenas cotas duales, y los planos de corte son una herramienta con alto potencial para enmendar estos problemas. Un primer acercamiento a este análisis sería la traducción de desigualdades conocidas para RSA sin protección a DPP y SBPP.

6.2.2. Descomposición de Benders

Los procedimientos basados en la descomposición combinatoria de Benders dieron los mejores resultados computacionales del trabajo. Sin embargo, hay aún mucho material para explorar relacionado a estas ideas. Por un lado, no experimentamos con distintas

funciones objetivo en el modelo maestro y auxiliar, más allá de pasar a devolver la primera solución factible hallada por el modelo auxiliar en un caso particular de instancias.

Por otro lado, sentimos que hay mucho potencial no explorado en la producción de cortes en cada iteración del algoritmo. Desde más estrategias de eliminación de asignaciones, hasta un análisis del ruteo desde el punto de vista de la teoría de grafos, es posible que haya ciertos criterios cuyo tiempo agregado valga la pena y reduzca el tiempo de ejecución total.

Consideramos también que puede ser de interés introducir las optimizaciones halladas para los modelos, como las heurísticas primales, a la descomposición. En particular, explotar el potencial visto de las heurísticas para acelerar la resolución de SBPP podría mejorar significativamente los tiempos de resolución de la fase de ruteo de la descomposición de dicho modelo, el cual es actualmente un gran cuello de botella.

6.2.3. Otras variantes de RSA

En este trabajo estudiamos la resolución de la versión *offline* de RSA, en la cual las demandas son sabidas de entrada. Un proyecto interesante a llevar a cabo sería aplicar las ideas desarrolladas en este trabajo a la variante *online*, en donde pueden llegar demandas en cualquier momento y se debe poder acomodarlas en la red rápidamente. Se estudiaría en este caso cuáles de las ideas presentadas dan buenos resultados y cuáles no resultan útiles.

Bibliografía

[1] Wassily W. Leontief. *The Structure of American Economy, 1919-1939: An Empirical Application of Equilibrium Analysis*. Harvard University Press, Cambridge, MA, 1941.

[2] Ralph E Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[3] Richard M Karp. Reducibility among combinatorial problems. *Complexity of computer computations*, pages 85–103, 1972.

[4] Ralph Gomory and RAND CORP SANTA MONICA CA. *An Algorithm for the Mixed Integer Problem: Notes Linear Programming and Extensions-Part 54*. Rand, 1960.

[5] Egon Balas. Disjunctive programming. *Annals of discrete mathematics*, 5:3–51, 1979.

[6] Egon Balas. Facets of the knapsack polytope. *Mathematical programming*, 8:146–164, 1975.

[7] Laurence A Wolsey. Faces for a linear inequality in 0–1 variables. *Mathematical Programming*, 8(1):165–178, 1975.

[8] Harlan Crowder, Ellis L Johnson, and Manfred Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.

[9] Tony J Van Roy and Laurence A Wolsey. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35(1):45–57, 1987.

[10] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1):295–324, 1993.

[11] Aleksandra Knapińska, Robert Kanimba, Yusuf Yesilyurt, Piotr Lechowicz, and Krzysztof Walkowiak. Application of ensemble regression methods in elastic optical network optimization. In *IEEE International Conference on Transparent Optical Networks (ICTON) 2023*, 04 2024.

[12] Ori Gerstel, Masahiko Jinno, Andrew Lord, and S. J. Ben Yoo. Elastic optical networking: a new dawn for the optical layer? *IEEE Communications Magazine*, 50, 2012.

[13] Konstantinos Christodoulopoulos, Ioannis Tomkos, and Emmanuel A Varvarigos. Elastic bandwidth allocation in flexible ofdm-based optical networks. *Journal of Lightwave Technology*, 29(9):1354–1366, 2011.

[14] Yang Wang, Xiaojun Cao, and Yi Pan. A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *2011 Proceedings IEEE Infocom*, pages 1503–1511. IEEE, 2011.

[15] Kostas Christodoulopoulos, Ioannis Tomkos, and Emmanouel A Varvarigos. Routing and spectrum allocation in ofdm-based optical networks with elastic bandwidth allocation. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–6. IEEE, 2010.

[16] Miroslaw Klinkowski and Davide Careglio. A routing and spectrum assignment problem in optical ofdm networks. In *First European Teletraffic Seminar*, 2011.

[17] Luis Velasco, Miroslaw Klinkowski, Marc Ruiz, and Jaume Comellas. Modeling the routing and spectrum allocation problem for flexgrid optical networks. *Photonic Network Communications*, 24:177–186, 2012.

[18] Mateusz Żotkiewicz, Michal Pióro, Marc Ruiz, Miroslaw Klinkowski, and Luis Velasco. Optimization models for flexgrid elastic optical networks. In *2013 15th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4. IEEE, 2013.

[19] Mirosław Klinkowski and Krzysztof Walkowiak. A simulated annealing heuristic for a branch and price-based routing and spectrum allocation algorithm in elastic optical networks. In *Intelligent Data Engineering and Automated Learning–IDEAL 2015: 16th International Conference, Wroclaw, Poland, October 14-16, 2015, Proceedings 16*, pages 290–299. Springer, 2015.

[20] Anliang Cai, Gangxiang Shen, Limei Peng, and Moshe Zukerman. Novel node-arc model and multiiteration heuristics for static routing and spectrum assignment in elastic optical networks. *Journal of Lightwave Technology*, 31(21):3402–3413, 2013.

[21] Hai Dao Thanh. *Contribution to Flexible Optical Network Design: Spectrum Assignment and Protection*. PhD thesis, Télécom Bretagne; Université de Bretagne Occidentale, 2014.

[22] Ajmal Muhammad, Marija Furdek, Georgios Zervas, and Lena Wosinska. Filterless networks based on optical white boxes and sdm. In *ECOC 2016; 42nd European Conference on Optical Communication*, pages 1–3. VDE, 2016.

[23] Federico Bertero, Marcelo Bianchetti, and Javier Marenco. Integer programming models for the routing and spectrum allocation problem. *Top*, 26(3):465–488, 2018.

[24] Youssouf Hadhbi, Hervé Kerivin, and Annegret Wagler. A novel integer linear programming model for routing and spectrum assignment in optical networks. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 127–134. IEEE, 2019.

[25] Yang Wang, Xiaojun Cao, Qian Hu, and Yi Pan. Towards elastic and fine-granular bandwidth allocation in spectrum-sliced optical networks. *Journal of Optical Communications and Networking*, 4(11):906–917, 2012.

[26] Emmanouel A Varvarigos and Konstantinos Christodoulopoulos. Algorithmic aspects in planning fixed and flexible optical networks with emphasis on linear optimization and heuristic techniques. *Journal of lightwave technology*, 32(4):681–693, 2013.

[27] Emmanouel A Varvarigos and Konstantinos Christodoulopoulos. Algorithmic challenges in flexible optical networks. In *2014 International Conference on Computing, Networking and Communications (ICNC)*, pages 236–241. IEEE, 2014.

[28] Mirosław Klinkowski, Mateusz Żotkiewicz, Krzysztof Walkowiak, Michał Pióro, Marc Ruiz, and Luis Velasco. Solving large instances of the rsa problem in flexgrid elastic optical networks. *Journal of Optical Communications and Networking*, 8(5):320–330, 2016.

[29] Mirosław Klinkowski and Krzysztof Walkowiak. Offline rsa algorithms for elastic optical networks with dedicated path protection consideration. In *2012 IV International Congress on Ultra Modern Telecommunications and Control Systems*, pages 670–676. IEEE, 2012.

[30] António Eira, João Pedro, and João Pires. Optimized design of shared restoration in flexible-grid transparent optical networks. In *Optical Fiber Communication Conference*, pages JTh2A–37. Optica Publishing Group, 2012.

[31] Mirosław Klinkowski. An evolutionary algorithm approach for dedicated path protection problem in elastic optical networks. *Cybernetics and Systems*, 44(6-7):589–605, 2013.

[32] Mirosław Klinkowski. A genetic algorithm for solving rsa problem in elastic optical networks with dedicated path protection. In *International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions*, pages 167–176. Springer, 2013.

[33] Ireneusz Olszewski and Ireneusz Szcześniak. The trade-offs between optimality and feasibility in online routing with dedicated path protection in elastic optical networks. *Entropy*, 24(7):891, 2022.

[34] Xu Shao, Yong-Kee Yeo, Zhaowen Xu, Xiaofei Cheng, and Luying Zhou. Shared-path protection in ofdm-based optical networks with elastic bandwidth allocation. In *Optical Fiber Communication Conference*, pages OTh4B–4. Optica Publishing Group, 2012.

[35] Aras Tarhan and Cicek Cavdar. Shared path protection for distance adaptive elastic optical networks under dynamic traffic. In *2013 5th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 62–67. IEEE, 2013.

[36] Krzysztof Walkowiak and Mirosław Klinkowski. Shared backup path protection in elastic optical networks: Modeling and optimization. In *2013 9th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 187–194. IEEE, 2013.

[37] Lu Ruan and Yanwei Zheng. Dynamic survivable multipath routing and spectrum allocation in ofdm-based flexible optical networks. *Journal of Optical Communications and Networking*, 6(1):77–85, 2014.

[38] Joy Halder, Tamaghna Acharya, Monish Chatterjee, and Uma Bhattacharya. On spectrum and energy efficient survivable multipath routing in off-line elastic optical network. *Computer Communications*, 160:375–387, 2020.

[39] Jin Y Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716, 1971.

[40] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[41] Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[42] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[43] Ayoub Insa Corréa, André Langevin, and Louis-Martin Rousseau. Scheduling and routing of automated guided vehicles: A hybrid approach. *Computers & operations research*, 34(6):1688–1707, 2007.

[44] Fabricio Oliveira, Ignacio E Grossmann, and Silvio Hamacher. Accelerating benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain. *Computers & Operations Research*, 49:47–58, 2014.

[45] Curtiss Luong. *An examination of Benders' decomposition approaches in large-scale healthcare optimization problems*. University of Toronto (Canada), 2015.

[46] Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10:237–260, 1972.

[47] John N. Hooker. *Logic-based Benders Decomposition*, volume 27 of *Applied Optimization*. Springer, 2000.

[48] Gianni Codato and Matteo Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[49] Michał Aibin and Krzysztof Walkowiak. Simulated annealing algorithm for optimization of elastic optical networks with unicast and anycast traffic. In *2014 16th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4. IEEE, 2014.

[50] Gang Feng, Christos Douligeris, and Miroslaw Klinkowski. A heuristic for routing, modulation and spectrum allocation in spectrum sliced elastic optical path network. In *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 111–115. IEEE, 2015.

Apéndices

Resultados Computacionales Adicionales

Los resultados computacionales de algunos de los experimentos que llevamos a cabo a lo largo del trabajo fueron presentados de manera abreviada para promover la legibilidad del documento. Como igualmente estos resultados son de interés y respaldan distintas afirmaciones que hicimos en sus respectivas secciones, detallamos a continuación un listado completo de los mismos.

1. DPP Con restricción de caminos titulares y de respaldo

Aquí reportamos los resultados que no incluímos en la sección 3.3.1.2 de este trabajo. Estos consisten en los Optimality Gaps y en las mejoras de tiempos de ejecución obtenidas al variar el parámetro k, que indica la cantidad de caminos candidatos a computar para cada demanda.

k = 20

Con un valor de $k = 20$, lo que observamos en la figura 3 es que si bien en obtiene mejoras en tiempos de ejecución para las instancias más grandes, estas mejoras no compiten con aquellas ofrecidas por el parámetro ganador $k = 10$, que se caracterizaba por brindar mejoras de mayor magnitud en un rango mucho más amplio de instancias. Cabe destacar que, como se puede ver en la figura 3 el rendimiento en términos del valor de la función objetivo con el warm start hallado es idéntico, con lo cual concluimos que las diferencias en tiempo se deben principalmente a que es más costoso resolver el modelo auxiliar cuando se tiene mayor libertad de elección para los caminos.

k = 40

Con este valor de k, lo que notamos es un buen rendimiento para instancias grandes, incluso en ciertos casos exhibiendo mejor desempeño que para el parámetro seleccionado $k = 10$. Mirando los Optimality Gaps reportados, se puede ver que estas mejoras corresponden en parte a que los warm starts producidos con $k = 40$ son de mejor calidad. Más aún, esta configuración logra obtener warm starts en ciertos casos puntuales en donde $k = 10$ no lo lograba, llevando a grandes mejoras de tiempo en estos casos.

No seleccionamos este valor como el superior, sin embargo, debido a los tiempos reportados con instancias bajas. Consideramos que aquí sería ideal seleccionar el valor de k

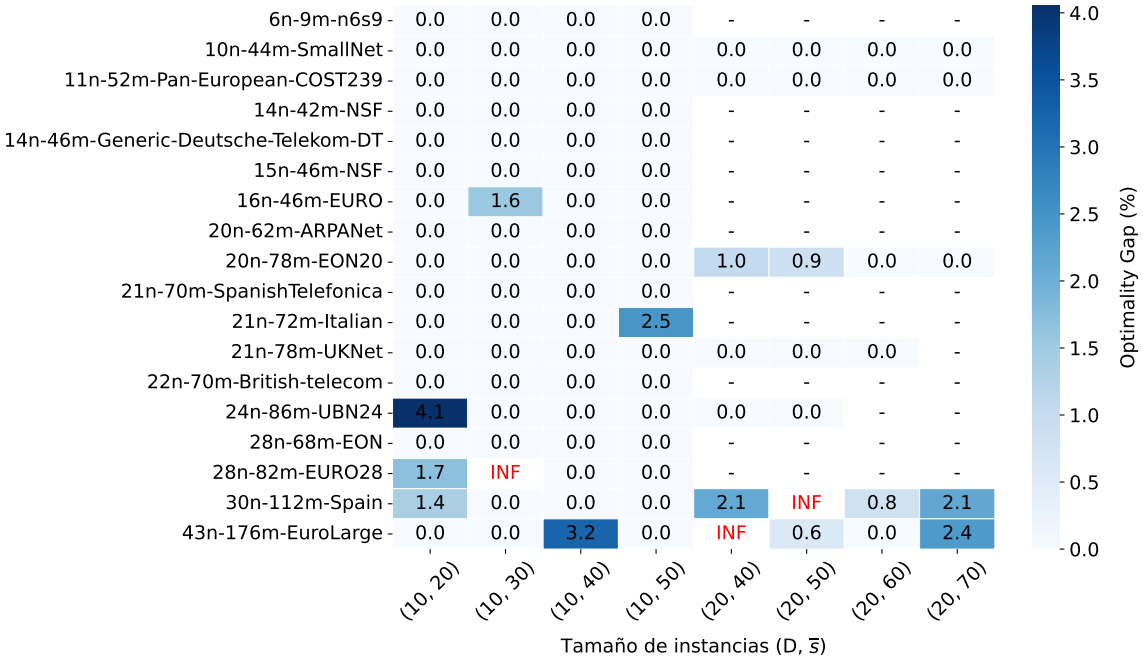

Figura 1: *Optimality Gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. Modelo: DPP. Caminos a fijar: titulares y de respaldo. k=20*

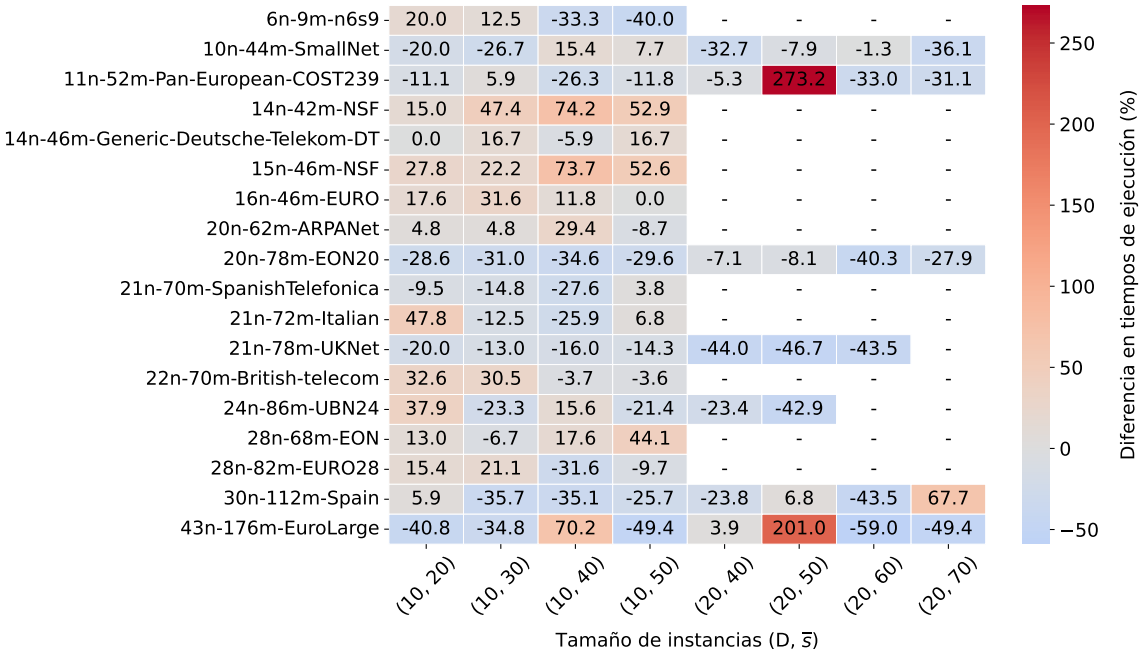

Figura 2: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. Modelo: DPP. Caminos a fijar: titulares y de respaldo. k=20*

a utilizar en base al tamaño de las instancias, empleando mayores valores cuando este lo requiera.

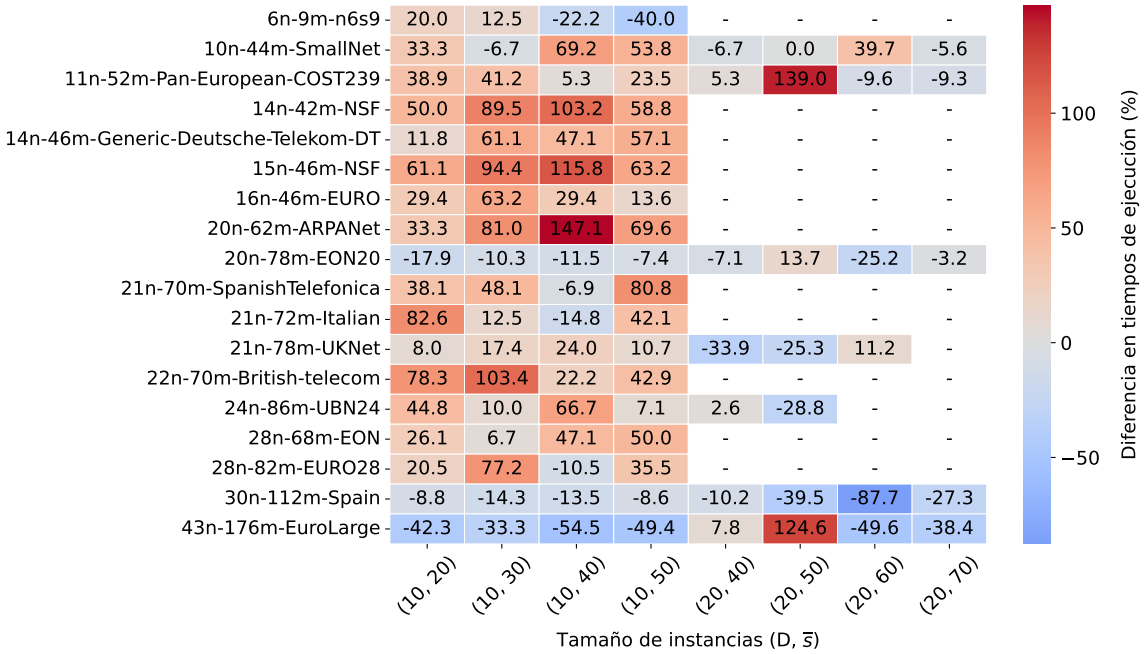

Figura 3: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. Modelo: DPP. Caminos a fijar: titulares y de respaldo. k=40*

2. SBPP Con restricción de caminos titulares

Esta sección contiene los resultados omitidos en la sección 3.3.2 de este trabajo. En dicha sección vimos que los mejores resultados se obtuvieron con $k = 5$. Este valor era suficiente para lograr la factibilidad del modelo reducido la gran mayoría de las veces, y en general conducía a warm-starts que mejoraban significativamente el tiempo total de obtención de soluciones para las instancias más grandes. A continuación detallamos qué se observó para otros valores de k.

k = 10

Podemos ver que con $k = 10$ el modelo reducido puede obtener soluciones óptimas en casi todas las instancias, lo cual presentaba dificultades con $k = 5$, donde muchas de las soluciones halladas (principalmente para instancias grandes) no alcanzaban el óptimo del problema completo. Sin embargo, el aumento en calidad de los warm-starts devueltos por el modelo reducido no conduce a una mejora notable en los tiempos de obtención de soluciones al problema completo; más bien se obtiene un escenario muy similar al caso $k = 5$, con pequeñas diferencias en instancias puntuales en los que a veces sale ganador este valor de k y a veces conviene el valor más chico.

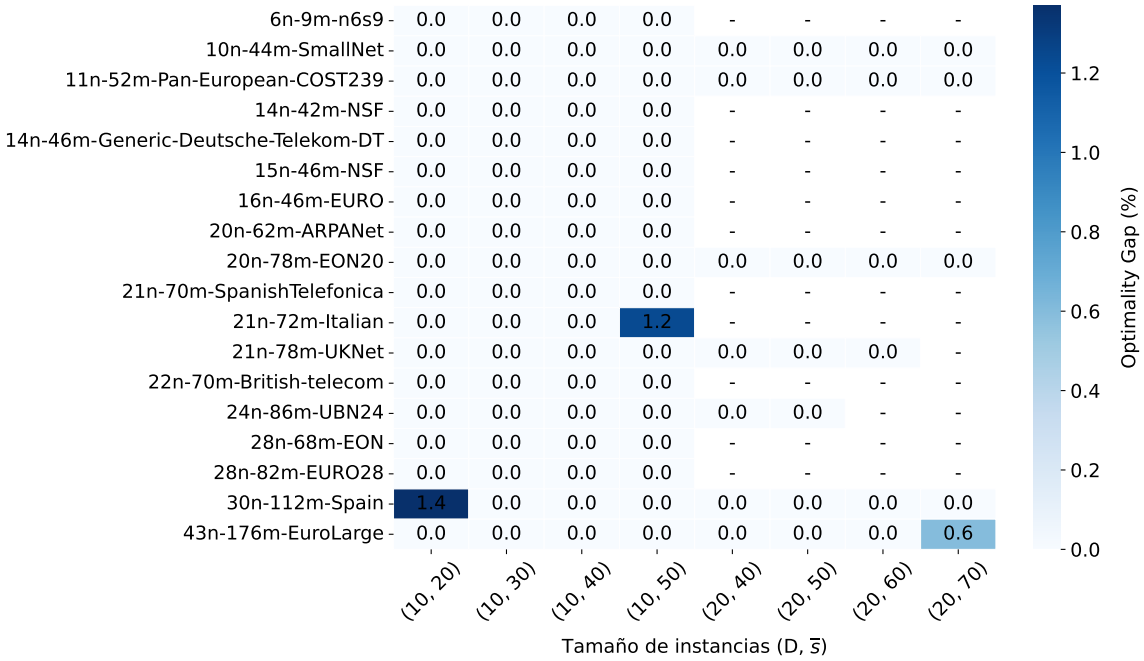

Figura 4: *Optimality Gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. Modelo: SBPP. Caminos a fijar: titulares. k=10*

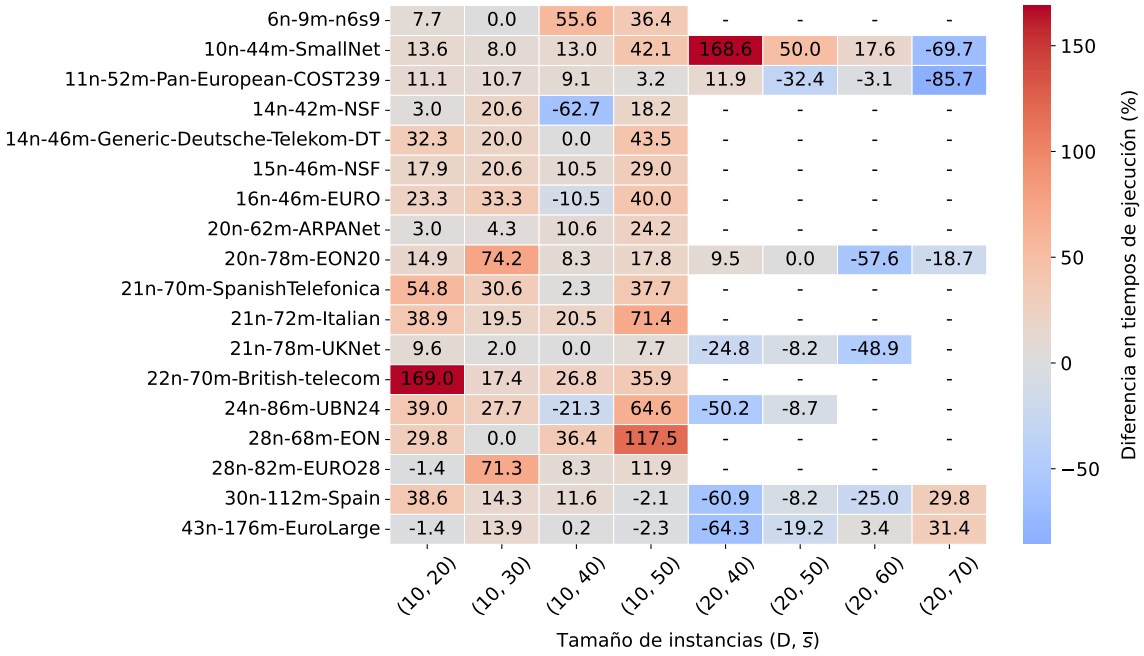

Figura 5: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. Modelo: SBPP. Caminos a fijar: titulares. k=10*

k = 20

Como se ve en la figura 6, fijar 20 caminos para cada demanda logra producir un modelo reducido que obtiene soluciones óptimas al problema completo para todas las instancias consideradas.

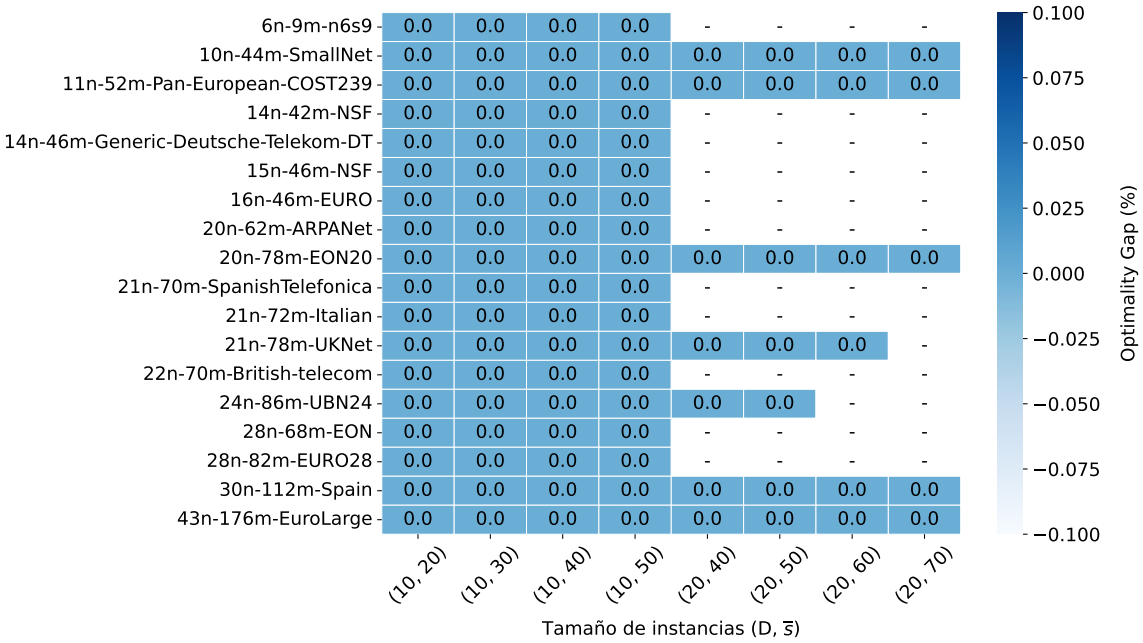

Figura 6: *Optimality Gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. Modelo: SBPP. Caminos a fijar: titulares. k=20*

Lo que se ve en 7, sin embargo, es que la certeza de factibilidad no conduce a mejoras en tiempos de ejecución. Similarmente al caso $k = 10$
, si bien se consiguen optimizaciones para las instancias más grandes, en general computar esta cantidad de caminos empeora el rendimiento del modelo.

3. SBPP Con restricción de caminos titulares y de respaldo

Los últimos resultados del apéndice tratan con los resultados que no incluímos en la sección 3.3.2.2 del trabajo. Recordemos que para esta tanda seleccionamos el valor $k = 10$ por ofrecer mejoras significativas en casi todas las instancias.

k = 20

Pasar a precomputar 20 caminos, de entre los cuáles el modelo deberá elegir uno a usar de titular y uno de respaldo, mejora notablemente la calidad de las soluciones halladas como se puede ver en la figura 8. Recordando que con $k = 10$ se reportaron muchas instancias no factibles, aquí se encuentra solo un caso en el que el modelo reducido no pudo hallar solución alguna. Además, en muchos casos las soluciones halladas se vuelven óptimas cuando antes no lo eran.

La razón por la cual no se eligió este parámetro para la cantidad de caminos es que

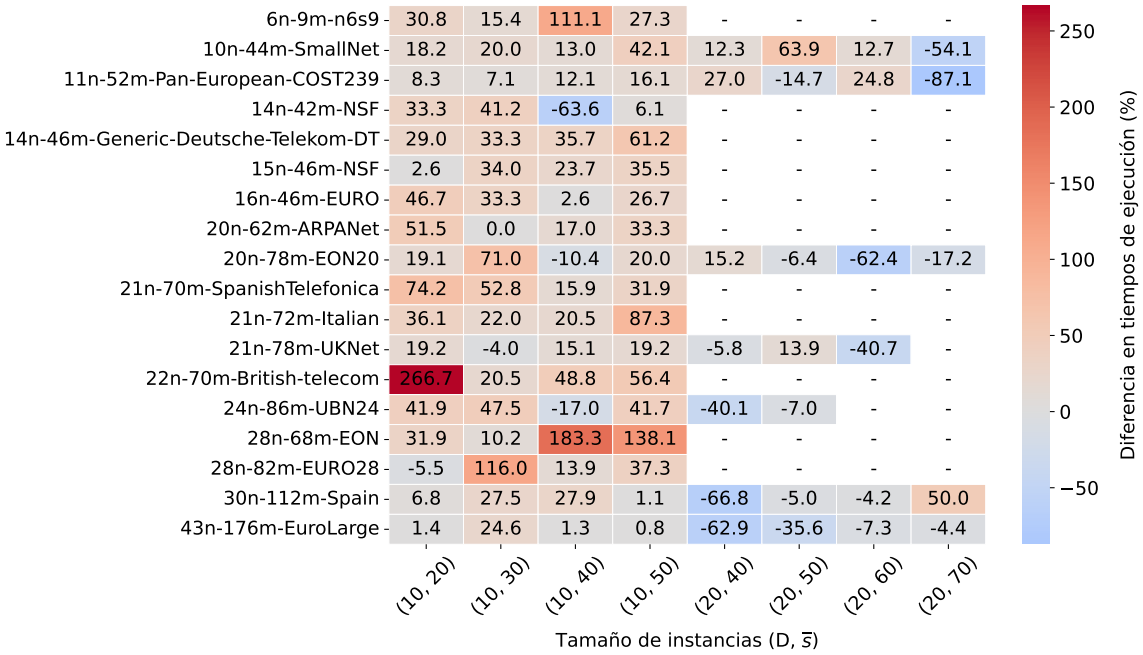

Figura 7: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. Modelo: SBPP. Caminos a fijar: titulares. k=20*

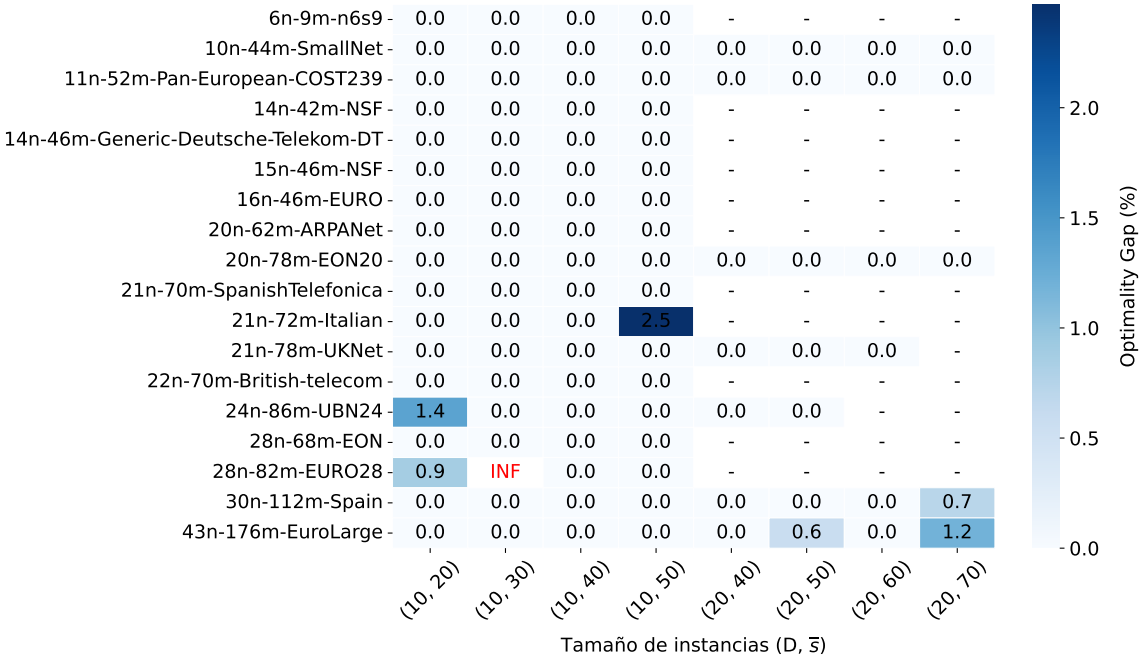

Figura 8: *Optimality Gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. Modelo: SBPP. Caminos a fijar: titulares y de respaldo. k=20*

este empeora significativamente el tiempo de ejecución para instancias chicas y no mejora los tiempos para instancias grandes. Es decir, perdemos las mejoras de carácter global que observábamos con $k = 10$ pero no ganamos nada que valga la pena. Esto se puede ver en la figura 9

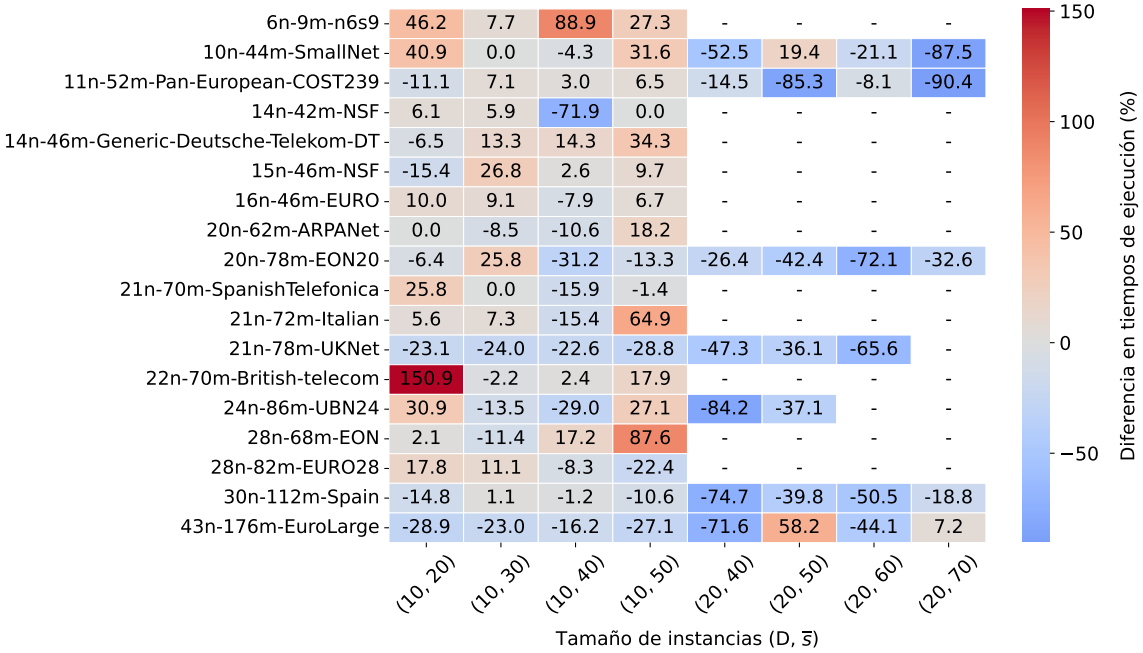

Figura 9: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. Modelo: SBPP. Caminos a fijar: titulares y de respaldo. k=20*

k = 40

Se puede ver en las figuras 10 y 11 que para $k = 40$ la tendencia se mantiene; observamos mejoras en la calidad de los warm-starts provistos por el modelo reducido, pero estos no tienen un impacto valioso en las instancias grandes (ya que valores más chicos de k ya conducían a soluciones óptimas en la mayoría de los casos), y por supuesto siguen empeorando el rendimiento en instancias chicas.

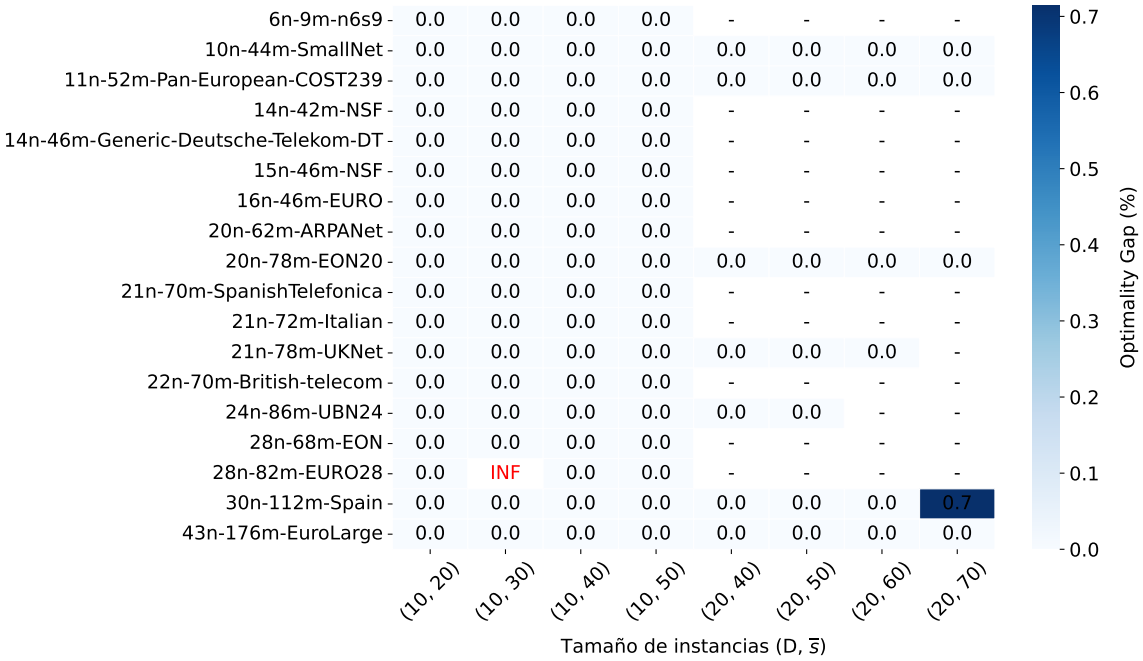

Figura 10: *Optimality Gap entre soluciones de modelo reducido y soluciones exactas, para un conjunto de instancias factibles. Modelo: SBPP. Caminos a fijar: titulares y de respaldo. k=40*

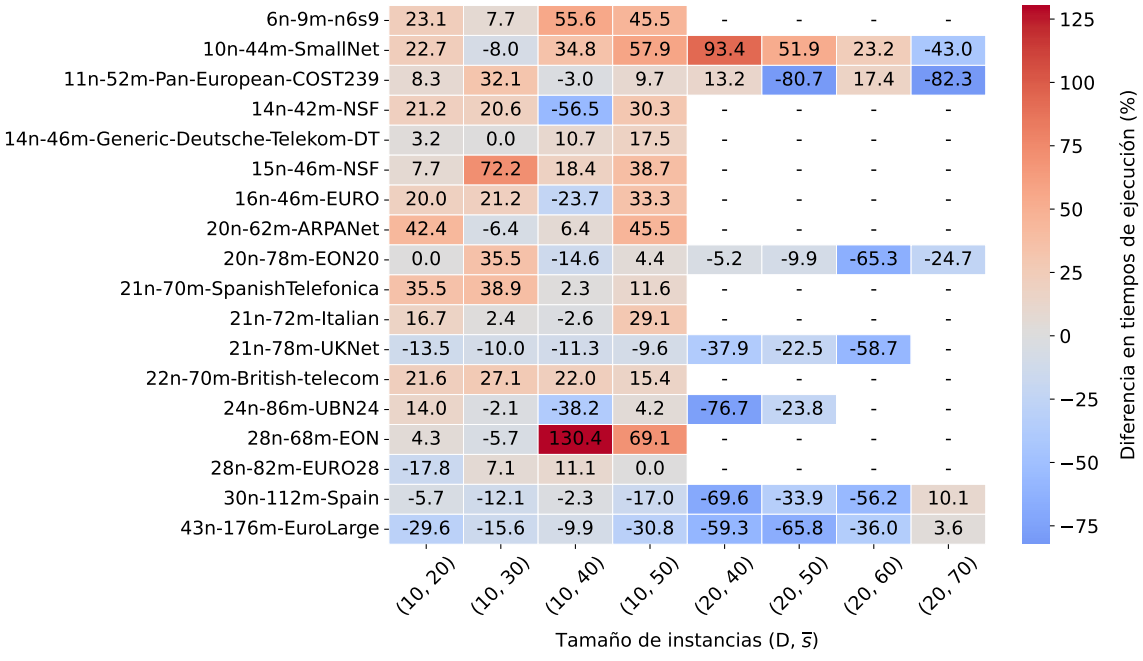

Figura 11: *Tiempo de resolución del modelo completo vs. Tiempo de la heurística + modelo completo con el warm-start provisto. Modelo: SBPP. Caminos a fijar: titulares y de respaldo. k=40*