Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico-Matemático

Posgrado en Ciencias con Orientación en Matemáticas

Un problema bi-nivel de cierre de instalaciones competitivas: Alternativas de modelación y solución.

por

Juan-Carlos García-Vélez

como requisito parcial para obtener el grado de

doctorado en ciencias con orientación en matemáticas

Marzo 2024

Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico-Matemático

Posgrado en Ciencias con Orientación en Matemáticas

Un problema bi-nivel de cierre de instalaciones competitivas: Alternativas de modelación y solución.

por

Juan-Carlos García-Vélez

como requisito parcial para obtener el grado de

DOCTORADO EN CIENCIAS CON ORIENTACIÓN EN MATEMÁTICAS

Marzo 2024

Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico-Matemático
Posgrado en Ciencias con Orientación en Matemáticas

Los miembros del Comité de Tesis recomendamos que la Tesis "Un problema bi-nivel de cierre de instalaciones competitivas: Alternativas de modelación y solución.", realizada por el alumno **Juan-Carlos García-Vélez**, con número de matrícula 1481090, sea aceptada para su defensa como requisito parcial para obtener el grado de Doctorado en Ciencias con Orientación en Matemáticas.

El Comité de Tesis

Dr. José-Fernando Camacho-Vallejo
Asesor

Dr. Diego Ruiz-Hernández
Co-Asesor

Dr. Juan A. Díaz-García
Co-asesor

Dr. Omar Jorge Ibarra Rojas
Sinodal

Dra. Dámaris Arizhay Dávila Soria
Sinodal

Vo. Bo.

Dr. Omar Jorge Ibarra Rojas
Coordinador del Posgrado en Ciencias
con Orientación en Matemáticas

San Nicolás de los Garza, Nuevo León Marzo 2024

Dedicatoria

A mis hijas Sofia y Mariel García e hijo Moises y mi querida esposa Alejandra Valenzuela.

A mis padres sr. Juan Carlos García y sra. Leticia Vélez

A mis hermanos Allison y Alan García

Agradecimientos

Esta tesis si bien ha requerido de esfuerzo y mucha dedicación por parte mía y del director de tesis, no hubiese sido posible su finalización sin la cooperación desinteresada de todas y cada una de las personas que a continuación mencionaré y muchas de las cuales han sido un soporte muy fuerte en momentos de angustia y desesperación. Gracias por brindarme su ayuda, sus conocimientos y su apoyo.

Agradezco a Alejandra Valenzuela ♥ por la motivación brindada para culminar la escritura de esta tesis y por darme dos hijas maravillosas y una familia increíble. Agradezco también a mis padres quienes han permitido trazar mi propio camino y caminar con mis propios pies. Ellos son mis pilares de la vida, les dedico este trabajo de titulación. Gracias Leticia Vélez y Juan Carlos García, padres los quiero y los amare por siempre.

Quiero agradecer especialmente a mi asesor, el Dr. Fernando Camacho por todo el escuerzo, tiempo y apoyo brindado a mi persona en ámbitos personales como profesionales desde que tengo el honor de conocerle. Muchas gracias por su paciencia. Sin su apoyo este trabajo no hubiera sido posible. Pero no solamente quiero agradecer por este período de tesis, sino por lo que antecede y trasciende a este período. Agradezco su apoyo y confianza en mi trabajo, por su paciencia y su capacidad para guiar, no solamente en el desarrollo de esta tesis, sino también en mi formación como investigador; por todas las cosas nuevas que aprendí y que no olvidaré. Por sus comentarios, sugerencias, consejos y críticas que se ven reflejados en este trabajo y en mi persona porque sirvieron de motivación. Por escucharme y aconsejarme cuando lo necesité y por tenerlo siempre como un soporte.

Les agradezco profundamente a mis co-asesores, el Dr. Juan A. Díaz por el tiempo que dedicó a este tema de tesis, por sus sugerencias e ideas y por los nuevos conocimientos que me transmitió, agradezco todas esas juntas virtuales en las que nos dedicábamos a programar y probar instancias. Agradezco también al Dr. Diego Ruíz-Hernández ya que él fue el precursor principal de la idea central de los problemas de cierre de instalaciones, por todos sus consejos y por todas las sugerencias e ideas para llevar a cabo este trabajo de tesis.

Agradezco a mis revisores Dr. Jorge Omar Ibarra y Dra. Dámaris Arizhay Dávila por sus certeras observaciones y valiosas sugerencias en la construcción y mejora del trabajo y por todo su tiempo invertido en la revisión de esta tesis.

Por último, pero no menos importante agradezco a la Universidad Autónoma de Nuevo León (UANL), a la Facultad de Ciencias Físico-Matemáticas (FCFM) y al Centro de Investigación en Ciencias Físico-Matemáticas (CICFIM) por todo el apoyo proporcionado durante este periodo de estudios. Además, es un placer para mí expresar mi gratitud al entonces Concejo Nacional de Ciencia y Tecnología (CONACYT) por la beca de apoyo.

A todos ustedes, Muchas Gracias.

Resumen

En este trabajo de tesis se presentan algunos enfoques de modelación y solución para un problema de cierre de instalaciones en ambientes competitivos modelado con programación bi-nivel previamente definido en mi tesis de maestría (ver García-Vélez (2019)). En dicho trabajo se modela una situación en donde se consideran dos empresas que ofrecen un servicio requerido por un conjunto de clientes. Dichas empresas compiten entre sí y deben cerrar algunas de sus instalaciones que mantienen operando con normalidad. Debido a la naturaleza de ambas empresas y al posicionamiento que tienen actualmente en el mercado, existe una jerarquía de mando entre ellas. Al momento de tomar la decisión sobre cuáles instalaciones van a cerrar, deben tener en cuenta las posibles acciones de la empresa rival y la reacción de los clientes. Este se conoce como "el problema $\overline{(r|p)}^{\mathcal{L}} - Centroide$", en donde en el nivel superior la empresa líder cierra sus instalaciones y en el nivel inferior, la empresa seguidora cierra las suyas de las instalaciones que tienen operando con normalidad. Entonces, este problema consiste en decidir las p y r instalaciones que van a cerrar ambas empresas. El símbolo $\mathcal{L}$ indica que en el modelo, la asignación de los clientes se hace tomando en cuenta un radio de lealtad y no en base a la distancia más pequeña, tales restricciones se agregan de manera válida en el problema del seguidor.

Para resolver el problema se desarrollan e implementan dos metaheurísticas: un procedimiento de búsqueda voraz adapatativo y aleatorizado (GRASP, por sus siglas en inglés) y una híbrida que combina GRASP con búsqueda tabú (TS por sus siglas en inglés) denominada GRASP-TS. Para evaluar el desempeño de los algoritmos propuestos se utilizan un conjunto de instancias de prueba del problema y las soluciones obtenidas se comparan con las soluciones óptimas, que se obtienen utilizando el método exacto para problemas de optimización bi-nivel propuesto por Fischetti et al. (2017) (a manera de tutorial se incluye un anexo en el cual se muestra la forma de utilizar dicho algoritmo). Los resultados numéricos muestran que los métodos de solución desarrollados en este trabajo se comparan favorablemente, ya que para la mayoria de las instancias de prueba se obtiene soluciones de buena calidad (soluciones óptimas en la mayoría de los casos) con un esfuerzo computacional menor al requerido por el algoritmo exacto. Además, se incluye una modelación equivalente

al problema bajo estudio, pero considerando las preferencias de los clientes, el cual resulta ser más robusto y atractivo pues elimina un gran número de variables y restricciones al modelo donde se considera la lealtad lo cual impacta en tiempo de ejecución como se puede observar a partir de la experimentación.

Palabras clave: Cierre de instalaciones, Radio de lealtad, Metaheurísticas, GRASP, TABÚ SEARCH, Preferencias.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En esta tesis doctoral nos enfocaremos en estudiar un problema competitivo de cierre de instalaciones considerando lealtad de los clientes. Primero, se brindará un panorama claro de la situación bajo estudio y las líneas de investigación desarrolladas en este trabajo doctoral, junto con la motivación, justificación y objetivo central de la misma. También se describe la metodología utilizada. Por último mencionamos la estructura de la tesis.

1.1 Descripción del problema

En los últimos dos años, diversos sectores tales como el social, político, educativo y turístico, entre otros, han sufrido dificultades económicas debido a la pandemia mundial por el COVID-19, ya que disminuyó drásticamente la demanda de servicios o porque sus instalaciones podrían considerarse fuentes importantes de contagio del virus. En esos casos, existen algunas empresas con franquicias para las cuales no resulta rentable mantener operando todas las sucursales y se ven en la necesidad de cerrar algunas de las que mantienen operando con normalidad. Por ejemplo, algunas empresas que cerraron instalaciones de manera permanente o temporal durante la pandemia en España son los bancos Santander y BBVA Bancomer[1], en México la aerolínea Aeroméxico, las tiendas departamentales Sears, Sanborns y Liverpool [2], la tienda de electrónicos Best Buy[3], entre otras[4].

Como se puede apreciar de los ejemplos descritos, cerrar instalaciones de las empresas que están operando con normalidad es un problema que recientemente se

[1] www.heraldo.es
[2] www.elsoldemexico.com.mx
[3] www.elfinanciero.com.mx
[4] www.forbes.com.mx

ha vuelto muy común. Es por eso que, en este trabajo doctoral, se investiga el caso de dos empresas que se dedican al mismo sector económico y ofrecen el mismo servicio en el mercado. Ambas empresas tienen instalaciones actualmente operando, es decir, cuentan con sucursales que atienden a un mercado específico de clientes en un área geográfica determinada. Debido a que los clientes ya están siendo atendidos por una de las dos empresas, estos tienen cierto grado de lealtad o preferencia sobre la empresa que le brinda el servicio. Sin embargo, ambas empresas se ven en la necesidad de cerrar algunas de sus instalaciones. Al hacer esto, es posible que pierdan algunos clientes y consecuentemente, habrá una pérdida de demanda. Por lo tanto, tienen que decidir cuales instalaciones cerrar de tal manera que maximicen sus ganancias, esto es, buscan ceder la menor cantidad de demanda de los clientes a la competencia. Se asume que los clientes tienen cierto grado de lealtad, entonces están dispuestos a recorrer una distancia máxima predeterminada para recibir el servicio de alguna otra sucursal abierta de las empresas de su preferencia. Como se mencionó anteriormente, esta situación fue propuesta y modelada en García-Vélez (2019) bajo un enfoque de programación bi-nivel, donde en el nivel superior la empresa con mayor poder adquisitivo (líder) cierra p instalaciones y la empresa del nivel inferior (seguidor) cierra r instalaciones. Para modelar la lealtad de los clientes se agregaron las restricciones correspondientes manteniendo el mismo número de variables que en el problema donde no se considera el supuesto de lealtad. Es decir, no se incluyeron nuevas variables de decisión, sino que solamente se incluyó un número polinomial de restricciones.

En esta tesis doctoral se continua con la investigación previa desarrollando las siguientes líneas de investigación:

* Se llevó a cabo un análisis de sensibilidad que muestra el impacto del porcentaje de captura de demanda de las empresas bajo ciertos criterios o niveles de lealtad por parte de los clientes. Esto se hizo con el fin de mostrar la importancia de considerar lealtad en este problema.

* Se adaptaron algunas instancias conocidas para resolver el problema con el método exacto propuesto por Fischetti et al. (2017).

* El modelo se valida con un caso práctico de dos empresas competidoras situadas en Monterrey, México.

* Se proponen dos metaheurísticas para resolver el problema: un GRASP y un híbrido GRASP-TS. Los resultados muestran que las instancias de prueba son resueltas de manera eficiente y a bajo costo computacional. Además, se resuelven instancias más grandes que las que se conocían en la literatura.

* Se propone un modelo equivalente, el cual está analizado bajo el enfoque de un problema de localización de instalaciones competitivas con preferencias de los clientes. El objetivo es buscar modelaciones más tratables para proponer futuros métodos de solución.

1.2 Motivación, justificación y contribución

La teoría de localización de instalaciones tiene como objetivo formular modelos matemáticos para identificar el mejor subconjunto de instalaciones para abrir de un conjunto de sitios potenciales en una región (Weber (1929)). Para ello, se deben cumplir ciertas restricciones inherentes a la situación particular bajo estudio. Este tipo de problemas pueden modelarse en un grafo, en un espacio continuo, o en un espacio discreto. En la mayoría de ellos se busca minimizar los costos de asignación, o bien, minimizar la distancia total de los clientes asignados a las instalaciones abiertas. Existe una amplia literatura sobre el tema, sin embargo, pocos autores se han interesado en el cierre de instalaciones competitivas, como ya lo hemos mencionado, en García-Vélez (2019) donde se propone un modelo de programación bi-nivel binario para hacer frente a esta problemática, el cual fue resuelto mediante un algoritmo de enumeración implícita de búsqueda en profundidad (depth-first search implicit enumeration) que resulta ser costoso computacionalmente hablando para instancias medianas y grandes. En la mayoría de las instancias de prueba se obtuvieron los valores óptimos pero en un tiempo de ejecución grande. Además, no se hicieron pruebas con instancias de mayor tamaño. Por lo tanto, el uso de algoritmos metaheurísticos parece ser una buena opción para resolver el problema en un tiempo de cómputo razonable. En la literatura se pueden encontrar algoritmos metaheurísticos eficientes para problemas de localización de instalaciones (no competitivas) que han sido modelado con programación bi-nivel en donde se consideran dos niveles de decision, un nivel superior donde el tomador de decisiones busca localizar las instalaciones para maximizar la demanda cubierta y otro tomador de decisiones en el nivel inferior asigna los clientes a la instalación más preferida, ver por ejemplo Díaz et al. (2017), Marić et al. (2012), Casas-Ramírez et al. (2017), MirHassani et al. (2015), Casas-Ramírez and Camacho-Vallejo (2017), Casas-Ramírez et al. (2018).

Como se mencionó anteriormente, cerrar instalaciones de las empresas que están operando con normalidad es un problema que recientemente ha tomado mucha relevancia. Además, decidir cuales sucursales se han de cerrar no es un proceso fácil, ya que eso implica una pérdida de demanda y por lo tanto, perder clientes. Es por esto que es importante abordar dicha problemática con una herramienta de optimización para brindar soporte en el proceso de toma de decisiones. Más aún, aunque se trate de una decisión de tipo estratégico, debido al contexto del

problema se requiere de metodologías eficientes para tener una mejor perspectiva de la situación y poder tomar decisiones estratégicas de mercado. De hecho, en muchas ocasiones es conveniente tomar una decisión rápida para tener una perspectiva clara del comportamiento del mercado y para tener una idea del impacto de los clientes en la distribución final del mercado entre los diferentes competidores. Por ello, resulta de interés proponer metodologías que proporciones soluciones óptimas o soluciones cercanas a la solución óptima, con un esfuerzo computacional razonable.

Con el fin de destacar la importancia de considerar lealtad por parte de los clientes en un mercado competitivo al momento de decidir a cual instalación asignarse y medir la afectación en el porcentaje de demanda capturada por las empresas, se lleva a cabo un análisis de sensibilidad bajo ciertos niveles de lealtad por parte de los clientes, la cual nos indica la importancia de considerar dicha característica en este tipo de modelos competitivos.

En la literatura existen métodos exactos que aseguran el valor objetivo óptimo y también metaheurísticos que si bien no aseguran obtener el valor óptimo logran obtener una buena solución a bajo costo computacional (ver, Alekseeva et al. (2015) y Davydov et al. (2014b)) para problemas bi-nivel de localización de instalaciones donde el nivel inferior corresponde a los clientes, quienes optimizan sus preferencias hacia las instalaciones abiertas. Inspirados en esa idea, planteamos una versión equivalente del problema de cierre de instalaciones con lealtad pero modelado como un problema bi-nivel de localización de instalaciones con preferencias.

Por otra parte, se sabe que el comportamiento de los mercados es muy volátil y cambia conforme pasa el tiempo pues no se puede saber cuando va a ocurrir alguna recesión económica en algún país determinado o bien alguna pandemia como la generada por el COVID-19, las cuales son situaciones que pueden afectar la demanda de los clientes, es por ello que nos surge la necesidad de proponer el modelo de cierre de instalaciones con incertidumbre en las demandas.

1.3 Objetivo

El objetivo principal de esta tesis es estudiar a profundidad un problema bi-nivel de cierre de instalaciones competitivas considerando la lealtad de los clientes y proponer metodologías de solución que brinden soluciones de buena calidad a un bajo costo computacional. Además, de destacar la importancia de considerar la lealtad de los clientes, lo cual se muestra mediante un análisis de sensibilidad de los radios de lealtad. Otro de los principales objetivos es el de proponer un modelo de localización de instalaciones competitivas con preferencias de los clientes, el cual

resulta ser equivalente al problema bajo estudio.

1.4 Metodología

Para que esta tesis pudiera realizarse de manera satisfactoria se siguieron los siguientes pasos como parte de la metodología implementada.

1: Se sometió en una revista indexada el trabajo desarrollado en la tesis García-Vélez (2019), donde además se agregó un caso de estudio.

2: Se trabajó en los cambios y/o sugerencias que solicitaron los revisores se realizo un análisis de sensibilidad y se llevo a cabo una discusión sobre las funciones de utilidad en nuestro modelo. Además, se comparó nuestro algoritmo exacto con otro método exacto genérico de la literatura (ver Fischetti et al. (2017)), por lo que nuestras instancias de prueba fueron adaptadas para ser resueltas por este algoritmo más reciente.

3: También los revisores sugieren hacer un cambio en nuestras restricciones de considerando lealtad y nace una nueva formulación para nuestro problema. Se propuso un modelo equivalente de localización con preferencias de los clientes.

4: Se publica nuestro articulo García-Vélez et al. (2024) con los tres pasos mencionados previamente.

5: Se hizo una revisión extensa de literatura sobre problemas bi-nivel de localización de instalaciones competitivas.

6: Se hizo una revisión de literatura extensa sobre problemas de cierre y restructuración de instalaciones determinísticos.

7: Se hizo una extensa revisión de literatura sobre los problemas de localización y/o cierre de instalaciones que han utilizado métodos exactos, heurísticos y/o metaheurísticos para resolverlos. Se resumen las diferencias de cada uno de ellos con el trabajo realizado en esta tesis en una tabla mostrada en el capítulo 2.

8: Se desarrolló e implementó una metaheurística GRASP con el fin de resolver el problema.

9: Con el fin de poder comparar la eficiencia del primer método de solución Se desarrolló e implementó una segunda metaheurística híbrida GRASP-TS para resolver el problema.

10: Se lleva a cabo el análisis estadístico correspondiente de los resultados obtenidos con ambas metaheurísticas y se comparan los resultados con los del método exacto propuesto por Fischetti et al. (2017).

11: Se escribió un artículo donde se muestran los resultados obtenidos con ambas metaheurísticas.

12: Presentación de los avances de tesis en la XXVIII Escuela Nacional de Optimización y Análisis Numérico (ENOAN 2019).

13 Se presenta otro avance en el VIII Congreso de la Sociedad Mexicana de Investigación de Operaciones (CSMIO 2019).

14: Se participó en el seminario del Posgrado en Ciencias con Orientación en Matemáticas de la FCFM-UANL 2020.

15: Se expone y participa en el congreso internacional EURO 2021: 31st European Conference on Operational Research.

16: Por último, se presenta el avance en el IX Congreso Nacional de la Sociedad Mexicana de Investigación de Operaciones (CSMIO 2021).

1.5 Estructura

En este Capítulo 1 se presenta la descripción del problema bajo estudio y se da a conocer el objetivo central de la tesis. También se describe la justificación de la investigación y la motivación que se tuvo para realizarla. Por último, se detalla la metodología propuesta.

En el Capítulo 2 se presenta una extensa revisión de literatura sobre los problemas de localización y/o cierre de instalaciones competitivas donde se utilizan métodos exactos, heurísticos y metaheurísticos para resolverlos. Además, al final de la sección se mencionan los problemas de cierre o restructuración que consideran incertidumbre.

En el Capítulo 3 se presenta la formulación matemática correspondiente al problema de cierre de instalaciones competitivas con lealtad de los clientes definido como en García-Vélez (2019). Primero, se define el modelo sin lealtad de los clientes. Se inicia el capítulo con los conceptos importantes sobre la programación bi-nivel. Además, se muestra un modelo equivalente al problema de cierre de instalaciones competitivas pero visto como un modelo de localización de instalaciones con preferencias de los clientes, se hace una analogía sobre el comportamiento de los conjuntos de lealtad y la matriz de preferencias.

En el Capítulo 4 se da una breve introducción a los métodos exactos y se describen a detalle los dos métodos exactos que se utilizaron para resolver los problemas bajo estudio.

En el Capítulo 5 se proponen dos metaheurísticas de solución para resolver el problema con lealtad presentado en el capítulo 3. Se propone un GRASP y con la finalidad de intensificar la búsqueda se propone una híbrida que combina el GRASP con búsqueda tabú, denominada GRASP-TS.

En el Capítulo 6 se presenta la experimentación computacional al resolver los problemas bajo estudio mediante los métodos exactos y metaheurísticos descritos en los dos capítulos previos para un conjunto grande de instancias y se describen los resultados obtenidos.

En el Capítulo 7 se lleva a cabo el análisis de sensibilidad que nos muestra la importancia de considerar lealtad en nuestro problema. Se ve el impacto en el porcentaje de captura de las demandas si se varían los valores umbrales de los radios de lealtad.

Por último, en el Capítulo 8 se presentan las conclusiones realizadas producto de la investigación hecha en esta tesis doctoral y se presentan las extensiones naturales del trabajo a futuro.

Capítulo 2

Revisión de literatura

En la sección anterior se mencionó que estamos considerando un problema donde dos empresas que se dedican al mismo sector económico se ven en la necesidad de cerrar algunas de las instalaciones que mantienen operando con normalidad. Ambas empresas brindan un servicio específico a sus clientes. Dicha situación se aborda desde un punto de vista competitivo donde las empresas buscan perder la menor cantidad de demanda considerando lealtad de los clientes. Debido a eso, en esta revisión de literatura nos enfocaremos primero en mencionar aquellos problemas de cierre de instalaciones seguido de los problemas de apertura de instalaciones, todos ellos modelados en espacios continuos, discretos o en una red. Se verifica si existe competencia o no; y se pone principal atención en las diferentes reglas de asignación de clientes. Por ejemplo, si es proporcional con alguna función en especial (esto es, alguna función de utilidad), en base a la distancia más corta, o mediante lealtad. También, se hizo énfasis en los diferentes enfoques de solución que se adoptaron para resolver los problemas, es decir, si se utilizó un método exacto, un algoritmo heurístico o una metaheurística. Por último, se clasificó el tamaño de las instancias. Las pequeñas abren o cierran p y r instalaciones que se encuentren en los rangos $1 \leq p = r \leq 10$. De manera análoga para las medianas y grandes con valores de $10 \leq p = r \leq 20$ y $p = r \geq 20$, respectivamente. Todos estos trabajos se resumen al final de esta sección en la Tabla 2.1.

2.1 Cierre de instalaciones.

Existen apenas unos cuantos trabajos en la literatura sobre situaciones en las que se requiere cerrar instalaciones en el mercado o bien reestructurar su red de instalaciones, por ejemplo, en Wang et al. (2003) se modela una situación en la cual existen instalaciones que ya no conviene mantenerlas abiertas. Esto se puede deber al

cambio en la distribución de la demanda, a la distribución de la población en ciertas zonas geográficas, o simplemente por la baja demanda en las instalaciones actuales. Por esto, se ven en la necesidad de cerrar algunas de las instalaciones que se tienen operando con normalidad, pero también van a explorar la posibilidad de abrir algunas otras de manera simultánea con el fin de brindar un mejor servicio a sus clientes. Se modela el problema como uno de un solo nivel y demuestran que es NP-duro. Para obtener soluciones factibles se utilizan tres algoritmos heurísticos: un intercambio greedy, una búsqueda tabú y uno basado en una aproximación de relajación Lagrangiana. Para comparar los resultados obtenidos con los algoritmos propuestos, obtienen la solución óptima mediante CPLEX, y los resultados son buenos pues estos se obtienen con un esfuerzo de cómputo razonable incluso para instancias grandes. En ReVelle et al. (2007), se presentan dos problemas que estudian el cierre de las instalaciones debido a emergencias financieras. Uno de ellos investiga el cierre de las instalaciones en un entorno donde se considera la presencia de competidores, en el cual se busca ceder la cantidad mínima de demanda. El otro problema estudia el cierre de instalaciones en una situación de exigencia financiera, pero sin competencia. En Bhaumik (2010), se define la palabra deslocalización (delocation, en inglés) como el hecho de cerrar una instalación existente y no en el sentido de reubicar la instalación, como se conoce en el sector económico. Presenta un problema en el cual una empresa desea reducir el tamaño de su red de distribución debido a la baja demanda. Con esta decisión, se espera que el cierre de algunas instalaciones reduzca los costos y mejore el nivel de servicio. Se proporcionan dos modelos alternativos: el primero es una formulación con costos fijos que permite la reasignación de nodos de demanda entre las instalaciones restantes, y la segunda formulación (denominada modelo de deslocalización puro), en donde se agrega una restricción adicional de reasignación para que los clientes que estaban asignados en una instalación que se cerró, sean reasignados.

En Ruiz-Hernández et al. (2015), se presenta un nuevo modelo de programación binaria para la reestructuración de sucursales considerando capacidad. En ese trabajo se consideran los costos de cierre y los costos de las operaciones (a largo plazo). Se aborda el caso en que se puede cambiar el tamaño de las sucursales abiertas para mantener un nivel de servicio constante. También, se toma en cuenta la presencia de competidores y la oportunidad de ceder la participación en el mercado, lo cual ocurre cuando los costos de reestructuración sean muy elevados. Se obtienen buenos resultados, es decir, se obtienen buenas soluciones con un esfuerzo de computo razonable y se realiza un análisis de sensibilidad para identificar los parámetros críticos del modelo. Se llega a la conclusión de que el diseño final de la red depende de ciertas decisiones estratégicas relacionadas con la redundancia de las sucursales, así como su cercanía a los nodos de demanda y a las sucursales de la competencia.

En Ruiz-Hernández and Delgado-Gómez (2016), se presenta un problema de reducción de instalaciones bancarias con capacidades redundantes en una red después de la fusión y adquisición financiera. Se toma en cuenta el tamaño de las sucursales cerradas, la existencia de competidores, y la incertidumbre en la respuesta de la demanda. Además, se consideran dos tipos de clientes: (i) los que siguen con la misma institución bancaria después de la restructuración de la red y (ii) los que son cedidos a una institución bancaria diferente. Formulan el problema como un modelo de programación binaria estocástica de dos etapas y lo resuelven para tres escenarios diferentes. En el primer escenario el banco realiza la reestructuración completa durante la primera etapa y luego, durante la segunda etapa, observa la reacción de los clientes. En la primera etapa del segundo escenario, el banco solo decide cuáles sucursales cerrar y cuáles mantener abiertas. En la segunda etapa, se observa la reacción de los dos tipos de clientes y se realizan las modificaciones necesarias en las capacidades virtuales y/o físicas de la red. Por último, en la primera etapa del tercer escenario, el banco cierra las sucursales estratégicamente y re-ubica a los clientes a los que se les cerró su instalación y los cambios en las capacidades físicas de las sucursales abiertas se realizan en la segunda etapa.

En Ruiz-Hernández et al. (2017), se presenta un problema donde dos empresas pretenden cerrar instalaciones que están operando en el mercado. Se toman en cuenta las posibles acciones de sus empresas rivales y la reacción de los clientes afectados. El objetivo es minimizar la pérdida de participación en el mercado. Se presenta un modelo de deslocalización competitiva desde un enfoque de teoría de juegos. El resultado del juego se analiza para tres escenarios diferentes: cuando se considera un comportamiento voraz, con conjeturas de Cournot, y bajo estrategias Stackelberg. Se considera la existencia de ciertos niveles de lealtad por parte de los clientes hacia su proveedor original; esto se logra mediante una función de decaimiento lineal (decay function). Los resultados númericos consiguen obtener equilibrios de Nash cuando se realizan pequeñas reducciones en el tamaño de la red (menos del 20 % de instalaciones) bajo un comportamiento voraz. Para resolver instancias más grandes, se indica la necesidad de considerar las acciones de las otras empresas. Además, menciona la existencia de una ventaja en el juego Stackelberg si las empresas tienen un poder de mercado diferente. Con respecto al efecto de los diferentes niveles de lealtad del cliente, los experimentos numéricos confirmaron que como era de esperarse, cuanto menos leales son los clientes, mayor es la demanda capturada por la empresa más fuerte.

Por último, en García-Vélez et al. (2024) se aborda el mismo problema aquí considerado. Dicho problema tiene lugar cuando una empresa debe decidir cuales instalaciones cerrar para reducir el tamaño de su red de distribución, y debe minimizar la cuota de mercado que pierde. Cabe mencionar que el competidor también

lleva a cabo un proceso de reducción de su propia red. El problema se modela como un modelo de programación bi-nivel y se formula como una variante del problema $(r|p)-Centroide$. Definen al problema como el $(r|p)-Centroide$ negativo. La formulación asume una postura novedosa con respecto al comportamiento de los clientes. Particularmente, se considera la existencia de cierto nivel de lealtad por parte de los clientes, esto es, pueden estar dispuestos a asignarse una instalación más distante de su proveedor habitual antes de visitar una instalación más cercana del competidor, lo cual trata de modelar una práctica que tiene sentido en la vida real. Se propone un algoritmo de enumeración implícita de búsqueda en profundidad para resolver instancias medianas y grandes del problema. Además, para comparar los resultados los autores resuelven las mismas instancias utilizando un algoritmo genérico que resuelve problemas bi-nivel de programación entera mixta de forma exacta. Se realiza una extensa experimentación numérica para probar la eficiencia computacional de los algoritmos. Además, se plantea un caso de estudio basado en la vida real para mostrar la validez práctica del problema estudiado.

2.2 Localización de instalaciones.

Como hemos visto, solo unos cuantos autores se han interesado en abordar problemas de cierre de instalaciones, esto debido a que no se había tenido la necesidad de cerrar instalaciones que se encuentran operando con normalidad mismas que ofrecen un producto o servicio en el mercado. Es importante mencionar que dichos trabajos (salvo Ruiz-Hernández et al. (2017)) consideran la presencia de competidores, pero estos no influyen en la toma de decisiones de la empresa que va a reducir su red de demanda. Con el fin de tener un panorama más claro de cómo abordar y solucionar el problema bajo estudio se ha hecho una extensa revisión de literatura sobre problemas de localización de instalaciones competitivas, mismos que mencionaremos a continuación, los cuales nos indican que los investigadores han optado por los métodos exactos, heurísticos y metaheurísticos para resolver dichos problemas.

Antes de comenzar, es importante mencionar que los problemas de localización de instalaciones (FLP, por sus siglas en inglés) fueron definidos por primera vez en Weber (1929). Este tipo de problemas han sido frecuentemente estudiados y forman un área que sigue creciendo hasta el día de hoy. En general, el objetivo de esta área de investigación es formular modelos matemáticos para identificar el mejor subconjunto de instalaciones para abrir en una región, pero cumpliendo con ciertas restricciones inherentes a la situación bajo estudio. Este tipo de problemas pueden modelarse en un grafo, en un espacio continuo o en un espacio discreto y en la mayoría de ellos se busca minimizar los costos de asignación, o bien, minimizar la distancia total de

los clientes asignados a las instalaciones abiertas. Se recomienda ver Drezner and Hamacher (2001), Wolf (2011) y Laporte et al. (2019) para una visión más general sobre estos problemas de localización.

Un tipo especial de problemas de localización de instalaciones es cuando se consideran empresas competidoras y su respectiva captación de clientes en el mercado (CFL, por sus siglas en inglés). Básicamente, una empresa tiene que decidir donde abrir instalaciones sabiendo que una empresa competidora también abrirá las suyas. Por consecuencia, una de las empresas puede tener una pérdida en la demanda captada. Fue en Hotelling (1929) donde se introdujo ese concepto de competencia entre dos empresas para abrir instalaciones. El problema se estudió como un proceso de dos etapas. En la primera etapa, ambas empresas deciden la localización de las instalaciones, mientras que en la segunda etapa las empresas eligen los precios óptimos para los productos que se ofertaban.

Los problemas CFL se dividen en dos categorías principales, por un lado, está el problema $(r|X_p) - medianoide$ y el problema $(r|p) - Centroide$. A este último lo denotaremos como RPCP (ver Hakimi (1983)). Ambos problemas se modelan bajo el enfoque líder-seguidor para obtener un equilibrio de Stackelberg, lo cual da como resultado a modelos de programación bi-nivel. En el $(r|X_p) - medianoide$, el seguidor abre r instalaciones asumiendo que el líder tiene p instalaciones de un conjunto de sitios potenciales X_p. En el RPCP, el líder abre p instalaciones de un conjunto I de sitios potenciales sabiendo que el seguidor reaccionará abriendo r instalaciones de los $|I| - p$ sitios potenciales restantes. Ambas empresas toman sus decisiones tomando en cuenta la maximización de la demanda capturada.

En la literatura el RPCP se ha resuelto utilizando métodos exactos y procedimientos heurísticos y/o metaheurísticos. A continuación, mencionaremos las características principales de cada uno de estos métodos así como los documentos en los que se ha resuelto el RPCP mediante estas metodologías de solución.

2.2.1 Métodos exactos

Para comenzar, definamos método exacto como un procedimiento que considera y resuelve un problema de manera óptima, es decir, asegura encontrar la solución óptima. Existen tres tipos de métodos exactos comunes para resolver este tipo de problemas: los de ramificación y acotamiento (Branch and Bound), los planos de corte (en particular Branch and Cut) y la metodología de descomposición de Benders. Todos ellos de alguna manera resuelven relajaciones lineales para encontrar la solucion óptima. El primero de ellos consiste en hacer una enumeración de manera

inteligente para que no se tengan que examinar todas las combinaciones de soluciones posibles. En el método de ramificación y corte o más general en el método de planos cortantes lo que se busca es de alguna manera agregar restricciones lineales (llamadas cortes) al modelo matemático general, las cuales restringen porciones de la región factible del problema relajado y al resolver la relajación lineal a dicho problema la solución lineal toma valores enteros. Por último, la descomposición de Benders consiste en resolver dos problemas de manera iterativa: un problema denominado maestro y un problema dual que al ser resuelto se encuentra una cota superior al problema a resolver que en determinado momento dicha cota coincide con las funciones objetivo del problema maestro y por ende la solución óptima es encontrada.

Existen trabajos en los cuales han resuelto el RPCP de manera exacta, por ejemplo en en Alekseeva et al. (2010) se resuelve el problema utilizando un método exacto, en donde se obtienen cotas superiores para el problema al resolver una reformulación que resultaba en un programa entero mixto con un número exponencial de restricciones y variables con las cuales aseguran encontrar el valor óptimo, resuelven instancias de tamaño pequeño para este algoritmo pues $p = r = 5$ con 100 nodos de demanda.

En Roboredo and Pessoa (2013), se desarrolla un método exacto basado en ramificación y corte. Es importante notar que el algoritmo no está diseñado para resolver el modelo bi-nivel, sino a un modelo de un solo nivel con un número polinomial de variables, pero un número exponencial de restricciones. Además, implementan un problema de separación, el cual anula muchas de las restricciones, por lo que se obtienen cotas para el problema. Resuelven instancias de hasta 100 clientes con valores de $p = r = 15$.

En Alekseeva et al. (2015), se presenta una versión mejorada al método exacto que se propuso en Alekseeva et al. (2010), dicha mejora fue basada en la utilización de la reformulación de un solo nivel que se propone en Roboredo and Pessoa (2013).

2.2.2 Métodos heurísticos y metaheurísticos

Como hemos visto, aunque resolver problemas mediante métodos exactos asegura encontrar el valor óptimo, a veces esto es limitado debido a la alta combinatoria que esto requiere y solamente son capaces de resolver instancias a lo mucho de tamaño mediano. Con el fin de aumentar la eficiencia computacional en cuanto a tiempo de ejecución y además poder resolver instancias que requieran una mayor combinatoria, los investigadores han optado por implementar métodos heurísticos y metaheurísti-

cos que, aunque no aseguran encontrar la solución óptima estos logran obtener una muy buena solución a bajo costo computacional. A continuación, describimos dichos métodos. Un método heurístico es un conjunto de técnicas y/o reglas definidas por el usuario que explota las características de un problema en cuestión para encontrar soluciones de buena calidad a bajo costo computacional. Por otro lado, un método metaheurístico es similar a un método heurístico pero con la única diferencia que en ellos las reglas de implementación ya se encuentran bien definidas y establecidas por la comunidad científica y estas son adaptadas por el usuario al problema a resolver.

Existe una gran variedad de trabajos que han resuelto el RPCP mediante estos métodos, por ejemplo, en Bhadury et al. (2003) resuelven la versión continua del problema con dos algoritmos heurísticos, uno tipo greedy y otro de diferenciación mínima, los cuales muestran una rápida convergencia. En Carrizosa et al. (2012), se presenta una heurística alternada mejorada; en la cual primero generan una solución aleatoria del líder y después resuelven el problema del seguidor a optimalidad. Una vez hecho eso, se obtiene la solución óptima del líder y se presentan mejores resultados que en Bhadury et al. (2003) para el mismo conjunto de instancias. En Rodríguez et al. (2009), resuelven el mismo problema, pero utilizando un metaheurístico basado en el enjambre de partículas. Se consideran dos enjambres, uno para las instalaciones del líder y otro para el seguidor. Se obtienen buenos resultados para un conjunto de instancias que ellos generaron en el cuadrado unitario.

En Davydov et al. (2014a), utilizan una estructura de vecindario probabilística basada en intercambios de instalaciones. Este algoritmo era muy similar al desarrollado en Alekseeva et al. (2010). Para resolver el problema bi-nivel, se define un problema auxiliar para el seguidor, donde el líder quiere localizar una instalación adicional. El algoritmo no asegura encontrar el valor óptimo, sin embargo, en muchas de las instancias si se obtiene la solución óptima. En Davydov et al. (2014b), se utilizan dos metaheurísticas que se basan en la modificación del vecindario de búsqueda: un VNS y otra que ellos llaman búsqueda tabú estocástica, la cual utiliza el vecindario de intercambio probabilístico que implementó Alekseeva et al. (2010). En muchos casos, estos algoritmos resuelven en menor tiempo las mismas instancias que en Davydov et al. (2014a). En Campos-Rodríguez et al. (2012), se utiliza un algoritmo de enjambre de partículas con salto (JPSO, por sus siglas en inglés). Similar al PSO descrito arriba, aquí también se usan dos enjambres de partículas: uno para el líder y otro para el seguidor. Sin embargo, una de las principales diferencias es que las partículas saltan de un enjambre a otro dependiendo del vecindario y de la mejor posición obtenida para todas las partículas hasta ese momento.

En Biesinger et al. (2014), se presenta un algoritmo evolutivo con una búsqueda tabú incrustada para optimizar la localización del líder. Se realiza un archivo de soluciones para detectar soluciones candidatas que ya fueron visitadas y convertirlas

en soluciones aún no consideradas. Esto evita reevaluaciones innecesarias que consumen mucho tiempo, se reduce la convergencia prematura y se aumenta la diversidad de la población al mismo tiempo. En Biesinger et al. (2015) se propone un algoritmo genético híbrido con varias mejoras. Se archivaron las soluciones exploradas en estructuras de datos para reducir el número de evaluaciones de soluciones innecesarias y evitar la convergencia prematura, lo que conduce a una ganancia de eficiencia significativa. Otra parte importante del algoritmo es el procedimiento de mejora local integrado. Investigaron varias formas de combinar la búsqueda local con el registro de soluciones e identificaron el vecindario reducido para un mejor funcionamiento en la práctica. Consideraron diferentes métodos de evaluación de soluciones y encontraron una forma efectiva de combinarlos, lo que llevó al esquema de evaluación multinivel. Por último, mejoraron los resultados del algoritmo utilizando una búsqueda tabú en la mejora local.

Se han considerado versiones del RPCP modeladas en redes o grafos, por ejemplo, en Benati and Laporte (1994) quienes utilizan una búsqueda tabú para resolver el problema RPCP y además muestra la eficiencia de su metaheuristica al también resolver un problema $(r|X_p) - medianoide$ donde se conoce previamente la localización del líder. En Kress and Pesch (2012a), se utiliza una heurística y definen el problema $(r|p) - Centroide$ en red con demandas en los vértices y en las aristas. En Serra and ReVelle (1994a) se propone un enfoque heurístico para una variante del RPCP discreto que se basa en la resolución repetida de un problema de captura máxima (MAXCAP). El MAXCAP es similar al problema del $(r|X_p) - medianoide$ con la diferencia de que es posible colocar una instalación en una de las ubicaciones del líder, con el supuesto de que la demanda capturada se comparte por igual entre los dos competidores. El algoritmo se basa en una búsqueda local con un vecindario que tiene la estructura dada por intercambios. Las soluciones candidatas se evalúan resolviendo el MAXCAP por medio de un programa entero o usando la heurística de búsqueda local para instancias más grandes. En Kochetov et al. (2013) se propone una metaheurística basada en la mejor estrategia de respuesta. Para una solución dada de un nivel, reformulan el problema resultante como un programa entero lineal y encuentran la solución óptima mediante un optimizador comercial. El algoritmo finaliza si alcanzan un equilibrio de Nash o encuentran la solución previamente visitada. Los experimentos computacionales indican que el algoritmo toma un pequeño número de pasos y obtiene soluciones óptimas o casi óptimas. Para una revisión de literatura más específica del RPCP se puede consultar en Kress and Pesch (2012b).

Hemos visto que existen ciertas limitaciones para resolver el problema RPCP mediante un método exacto y también que existe una amplia literatura sobre trabajos que utilizaron métodos heurísticos y metaheurísticos para resolver el problema, para los cuales se obtuvieron buenos resultados e incluso algunos llegaron a encontrar

la solución óptima. Por lo tanto, es natural pensar que es buena idea implementar algunos de estos métodos para resolver el problema bajo estudio.

Como se mencionó anteriormente, en la siguiente tabla se muestran los trabajos descritos en esta revisión de literatura, donde en la primera columna se muestra el trabajo, después se clasifica el tipo de modelo como en red, continuo o discreto, la columna tipo indica si el modelo abre (A) o cierra (C) instalaciones seguido de la información que indica si el modelo aborda la competencia. También se indica el tipo de asignación si este es proporcional (PR), en base a distancia más corta (DC) o bien haciendo uso de la lealtad (L), se indica si el modelo fue resuelto usando un método exacto, una heurística o una metaheurística, columnas Ex., Heur., Meta., respectivamente. Por último, se indica si las instancias que resuelve cada trabajo son pequeñas (P), medianas (M) o grandes (G).

Clasificación de trabajos relacionados con esta investigación															
Referencia	**Modelación**			**Tipo**		**Competitivo**	**Asignación**			**Método de solución**			**Tamaño**		
	Red	**Cont.**	**Disc.**	**A**	**C**		**PR**	**DC**	**L**	**Ex.**	**Heur.**	**Meta.**	**P**	**M**	**G**
Hakimi (1983)	✓			✓		✓		✓							
Bhadury et al. (2003)		✓		✓		✓		✓			✓		✓	✓	✓
Carrizosa et al. (2012)		✓		✓		✓		✓			✓		✓	✓	
Rodríguez et al. (2009)		✓		✓		✓		✓				✓	✓		
Alekseeva et al. (2010)			✓	✓		✓		✓		✓	✓		✓	✓	
Alekseeva et al. (2015)			✓	✓		✓		✓		✓			✓	✓	
Davydov et al. (2014a)		✓		✓		✓		✓				✓	✓	✓	
Davydov et al. (2014b)			✓	✓		✓		✓				✓	✓	✓	✓
Campos-Rodríguez et al. (2012)			✓	✓		✓		✓				✓	✓		
Roboredo and Pessoa (2013)			✓	✓		✓		✓		✓			✓	✓	
Biesinger et al. (2014)			✓	✓		✓	✓					✓	✓		
Biesinger et al. (2015)			✓	✓		✓		✓				✓	✓	✓	✓
Benati and Laporte (1994)	✓			✓		✓		✓				✓		✓	✓
Kress and Pesch (2012a)	✓			✓		✓		✓			✓		✓		
Kochetov et al. (2013)	✓			✓		✓	✓					✓	✓		
Serra and ReVelle (1994a)	✓			✓		✓	✓				✓		✓		
Wang et al. (2003)			✓	✓	✓			✓				✓	✓	✓	
ReVelle et al. (2007)			✓		✓			✓							
Bhaumik (2010)	✓				✓			✓							
Ruiz-Hernández et al. (2015)			✓		✓	✓	✓			✓			✓		
Ruiz-Hernández and Delgado-Gómez (2016)	✓				✓	✓	✓				✓		✓	✓	
Ruiz-Hernández et al. (2017)	✓				✓	✓	✓				✓		✓	✓	
García-Vélez et al. (2024)	✓		✓		✓	✓			✓	✓			✓	✓	
Esta tesis	✓		✓		✓	✓			✓			✓	✓	✓	✓

Tabla 2.1: Clasificación de trabajos relacionados con esta investigación.

Capítulo 3

Modelación matemática bi-nivel: Problemas de cierre de instalaciones competitivas

En este capítulo primero se van a definir los conceptos básicos importantes sobre programación bi-nivel. Después, se definirán los problemas de cierre de instalaciones considerados en esta tesis, estos son, el modelo sin considerar lealtad de los clientes y después el modelo con lealtad de los clientes de manera similar que en García-Vélez (2019).

3.1 Programación Bi-nivel

En ocasiones, durante el proceso de optimización, los tomadores de decisiones se encuentran con decisiones interrelacionadas con una jerarquía. Aún más, algunas de estas decisiones no están en sus manos, es decir, existe una jerarquía de mando en la toma de decisiones, lo cual conduce a que la decisión adoptada por cada uno de los tomadores de decisiones afecta directamente la decisión de los otros tomadores de decisiones. Esto nos lleva a identificar que las decisiones se toman en diferentes niveles dentro de esta jerarquía. Una manera de abordar tales jerarquías es concentrarse en un nivel, pero incluyendo el comportamiento de los otros niveles como suposiciones. A esto se le conoce como programación multi-jerárquica o simplemente programación multi-nivel definida formalmente en Candler and Norton (1977).

Un caso particular de la programación multi-nivel es cuando se consideran solo dos tomadores de decisiones que van a tomar sus decisiones de manera jerarquizada, a dicha área de investigación se le conoce como programación en dos niveles o

programación bi-nivel.

De manera general, un problema de programación bi-nivel puede abordarse desde dos perspectivas: la teoría de juegos y la programación matemática. Desde el punto de vista de la teoría de juegos, hay dos tomadores de decisiones en competencia asociados con los problemas de nivel superior e inferior (en Stackelberg et al. (1952) se les llama líder y seguidor, respectivamente). En ese juego secuencial jerarquizado, primero el líder establece su estrategia, luego el seguidor reacciona racionalmente eligiendo su mejor estrategia teniendo un conocimiento completo de la decisión del líder. Además, se asume que se da información perfecta ya que el líder decidirá primero su estrategia y, después de eso, el seguidor decidirá su propia estrategia.

Desde el punto de vista de la programación matemática, un problema de programación bi-nivel consiste en un problema de programación matemática en el que las variables se pueden dividir en dos subconjuntos. El problema principal es que uno de los subconjuntos de variables debe estar determinado por la solución óptima de otro problema de programación matemática; es decir, un problema de programación bi-nivel es un problema de programación matemática donde una de sus restricciones es otro problema de optimización. A continuación, se muestra la formulación general presentada en Bard (2013).

Suponga que un primer tomador de decisiones, el del nivel superior denominado líder tiene control sobre el vector $x \in X \subseteq \mathbb{R}^n$ y que un segundo tomador de decisiones denominado seguidor tiene control sobre el vector $y \in Y \subseteq \mathbb{R}^m$. El líder toma primero una decisión y selecciona una x en un intento de minimizar $F(x, y(x))$, sujeto quizás a algunas restricciones adicionales. La notación $y(x)$ hace referencia al hecho de que el problema del líder está implícito en las variables del seguidor (por simplicidad se denotará a esta como y)). Esto se puede traducir de la siguiente manera: el seguidor observa las acciones del líder x y reacciona seleccionando una y para minimizar su función objetivo $f(x, y)$, sujeto a un conjunto de restricciones en las variables y para el valor particular de x elegido por el líder. Entonces podemos definir el siguiente modelo matemático:

$$\begin{aligned}
\min_{x \in X} \quad & F(x, y) && && (3.1)\\
\text{sujeto a:} \quad & G(x, y) && \leq \quad 0 && (3.2)\\
\min_{y \in Y} \quad & f(x, y) && && (3.3)\\
\text{sujeto a:} \quad & g(x, y) && \leq \quad 0 && (3.4)
\end{aligned}$$

donde $F, f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^l$, $G : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$, y $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$.

Las restricciones (3.1)-(3.3) definen el problema del líder, donde (3.1) es la función objetivo del líder quien busca minimizar la función $F(x, y)$, sujeto a las restricciones (3.2)-(3.3). Note que (3.3) denota otro problema de programación matemática llamado el problema del seguidor, el cual está definido por las restricciones (3.3)-(3.4), donde (3.3) busca minimizar la función $f(x, y)$ considerando las restricciones (3.4).

No está de más mencionar que el modelo presentado anteriormente es el caso más simple de un problema bi-nivel donde el problema del líder es un problema lineal y el problema del seguidor también. Este caso se conoce como el problema de programación bi-nivel lineal (BLPP, por sus siglas en inglés). En Bard (2013) se discute que este problema pertenece a la clase de complejidad NP-duro.

Para profundizar sobre este tipo de problemas, hay que tener en cuenta las siguientes definiciones importantes:

1. Conjunto de restricciones del problema de programación bi-nivel.

$$\Omega = \{(x, y) \mid G(x, y) \leq 0, g(x, y) \leq 0\}$$

2. Conjunto de soluciones factibles del seguidor para cada $x \in X$.

$$\Omega(x) = \{y \in Y \mid g(x, y) \leq 0\}$$

3. Conjunto de reacción racional del seguidor para cada $x \in \Omega(x)$.

$$\Psi(x) = \{y \in Y \mid y \ \textit{resuelve}: \ \text{mín}\{f(x, y) \leq 0 \mid y \in \Omega(x)\}\}$$

4. Región inducible.

$$IR = \{(x, y) \mid (x, y) \in \Omega, y \in \Psi(x)\}$$

Este último conjunto, denominado región inducible, es la región factible de un problema de programación bi-nivel. Es decir, en él se encuentran las soluciones factibles bi-nivel, que muchas veces puede llegar a ser no convexo o incluso vacío. Además, tome en cuenta que en $\Psi(x)$, el líder no puede sacar provecho sobre la decisión del seguidor para su propio beneficio, ya que este último tiene un comportamiento racional y toma una decisión conveniente para su propio beneficio. Por lo tanto, el líder no puede saber el verdadero valor de su función objetivo hasta conocer la respuesta del seguidor. Es aquí donde surge la gran incógnita: ¿Qué pasaría si para una decisión $\overline{x}$ del líder existen múltiples soluciones y en $\Psi(\overline{x})$, es decir, que pasa

con la función objetivo del líder en caso de que el seguidor tenga múltiples soluciones óptimas?

Es evidente que para todas esas soluciones $(\overline{x}, y) \in IR$ la función objetivo del seguidor toma el mismo valor, sin embargo, no es así para el líder. De ahí se derivan los dos enfoques de solución para problemas de programación bi-nivel: el enfoque optimista y el pesimista.

En el *enfoque optimista*, se asume que el líder tiene cierta influencia en la respuesta del seguidor, al hacer que este seleccione aquella solución $y \in \Psi(\overline{x})$ que sea mejor para el líder. Esto se puede apreciar mejor si asumimos lo siguiente:

$$\pi(y) := \min_{\overline{x}} \{F(\overline{x}, y) \mid y \in \Psi(\overline{x})\} \tag{3.5}$$

Note que la expresión (3.5) denota la función objetivo del líder mínima dado una solución $\overline{x}$ en donde se tienen múltiples soluciones óptimas para el seguidor. En ese sentido se puede definir el siguiente problema bi-nivel optimista:

$$\min_{y} \{\pi(y) \mid G(\overline{x}, y) \leq 0\}$$

Así vemos cómo esta enfoque optimista no parece tener ningún problema, a excepción en los casos donde exista la no cooperación, es decir donde el seguidor no actúa a favor del líder. Esto se conoce como el *enfoque pesimista* el cual busca perjudicar al líder. De manera análoga al caso optimista esto se puede ver de la siguiente forma:

$$\beta(y) := \max_{\overline{x}} \{F(\overline{x}, y) \mid y \in \Psi(\overline{x})\} \tag{3.6}$$

Así, el problema pesimista se reduce a:

$$\min_{y} \{\beta(y) \mid G(\overline{x}, y) \leq 0\}$$

3.2 Formulación Matemática

A continuación, abordaremos la situación bajo estudio que se definió en García-Vélez (2019). Consideramos el problema de una empresa que tiene como objetivo

reducir el número de instalaciones que tiene en un mercado determinado, minimizando la participación de mercado que se cede a su(s) competidor(es). Esta empresa interactúa jerárquicamente en el mercado con otra empresa, a la cual se refiere como un seguidor. La empresa ha decidido cerrar un número p de instalaciones y, para minimizar la pérdida de participación de mercado, debe tener en cuenta la reacción potencial de su competidor, quien pretende cerrar r instalaciones. Este problema está relacionado con el que enfrentan dos competidores que planean abrir una serie de instalaciones con el objetivo de capturar la participación máxima de mercado, que a menudo se conoce como el problema $(r|p)$-*Centroide*. Debido a esta similitud, nos referimos a nuestro problema de cierre de instalaciones jerárquicas como el problema $(r|p)$-*Centroide negativo*, denotado por $\overline{(r|p)}$-*Centroide*.

En la formulación matemática del problema $\overline{(r|p)}$-*Centroide*, se supone que los clientes son sensibles a la distancia, en el sentido de que si sus instalaciones actuales se cierran, se reasignarán a la instalación más cercana que permanece abierta. Sin embargo, es fácil encontrar ejemplos en los que los clientes permanezcan leales a su proveedor actual siempre que su instalación más cercana esté a su alcance. Esta situación se modela en términos de un *radio de lealtad*. La idea es que un cliente estará dispuesto a mantener su costumbre de satisfacer su demanda con cierta empresa siempre que la instalación abierta más cercana no esté más lejos que cierta distancia admisible (*radio de lealtad*). Esto ocurre independientemente del hecho de que haya instalaciones de un competidor dentro del mismo radio o no. Estas dos estrategias de asignación de clientes se tratan en las siguientes dos secciones 3.2.1 y 3.2.2, respectivamente.

3.2.1 El problema $\overline{(r|p)}-Centroide$

Considere un mercado competitivo que es dominado por dos empresas y están localizadas en un sistema discreto. La empresa con mayor poder adquisitivo, el líder, tiene una colección de instalaciones representadas con el conjunto I^L. Así mismo, la empresa seguidora tiene un conjunto I^F de instalaciones localizadas en diferentes puntos. El conjunto de todas las instalaciones en el mercado está representado por $I = I^L \cup I^F$. El conjunto de nodos consumidores o clientes viene dado por el conjunto J. El líder y el seguidor quieren cerrar p y r instalaciones, respectivamente, mientras mantienen tanto como sea posible su participación de mercado. Las variables de decisión y otros parámetros del modelo se introducen en las siguientes líneas:

PARÁMETROS:

b_j**:** Demanda del cliente $j \in J$;

d_{ij}**:** Distancia entre el cliente $j \in J$ y la instalación $i \in I$;

VARIABLES DE DECISIÓN:

x_i^L**:** Variable binaria que toma el valor de 0 si la instalación del líder $i \in I^L$ se cierra, y 1 en caso contrario;

x_i^F**:** Variable binaria que toma el valor de 0 si la instalación del seguidor $i \in I^F$ se cierra, y 1 en caso contrario;

y_j**:** Variable binaria que toma el valor de 0 si el cliente $j \in J$ es atendido por el líder, y 1 si es atendido por el seguidor.

Con la ayuda de algunas limitaciones en la asignación de clientes, el problema de cierre de instalaciones competitivas se puede formular como un problema de programación bi-nivel de la siguiente manera:

$$\max_{x^L} \quad \sum_{j \in J} b_j(1 - y_j) \tag{3.7}$$

$$\text{s.t.} \quad \sum_{i \in I^L} \left(1 - {x_i}^L\right) = p \tag{3.8}$$

$${x_i}^L \in \{0, 1\} \qquad \forall i \in I^L \tag{3.9}$$

donde x^F y y_j resuelven

$$\max \quad \sum_{j \in J} b_j y_j \tag{3.10}$$

$$\text{s.t.} \quad \sum_{i \in I^F} \left(1 - {x_i}^F\right) = r \tag{3.11}$$

$$y_j \leq \sum_{i \in I_j(x^L)} {x_i}^F \qquad \forall j \in J \tag{3.12}$$

$$x_i^F, y_j \in \{0, 1\}, \qquad \forall i \in I^F, \forall j \in J \tag{3.13}$$

Donde $I_j(x^L) = \{i \in I^F : d_{ij} \leq min_{k \in I^L | x_k^L = 1} d_{kj}\}$ representa el conjunto de todas las instalaciones del seguidor que están más cerca del nodo de demanda j que la instalación abierta más cercana del líder, para todo $j \in J$.

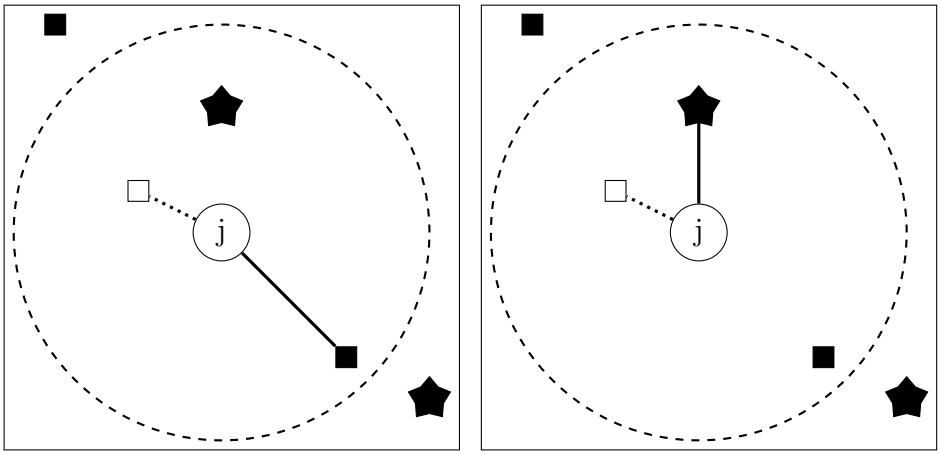
La función objetivo del líder (3.7) tiene como objetivo maximizar la demanda total retenida después de que ambas empresas hayan tomado sus decisiones de cierre. La restricción (3.8) indica que solo se cerrarán p instalaciones del líder. En el nivel inferior, el objetivo (3.10) del seguidor también es maximizar la demanda total capturada (o retenida). La restricción (3.11) indica el número de instalaciones a cerrar. Las restricciones (3.12) garantizan que los clientes serán asignados a la instalación más cercana. Finalmente, las restricciones (3.9) y (3.13) son las restricciones binarias estándar. Las ecuaciones (3.7)-(3.13) brindan una formulación de programación bi-nivel binaria para el problema $\overline{(r|p)}-Centroide$.

3.2.2 El problema $\overline{(r|p)}^{\mathcal{L}}-Centroide$

Con el fin de considerar un modelo más realista, se consideran que los clientes muestran cierto nivel de lealtad a su proveedor actual. Se introduce el término lealtad espacial para referirnos al tipo de lealtad a la marca que no se relaciona con un producto específico sino con la cadena o franquicia de un proveedor en particular, por ejemplo, los servicios bancarios, los restaurantes de comida rápida y la industria de la moda en las calles. La lealtad a una marca se ha explorado ampliamente en la mercadotecnia y en las ciencias del comportamiento (ver, por ejemplo, Tam et al., 2009; Nezakati et al., 2011; Rehman et al., 2012; Swoboda et al., 2013; Venter de Villiers et al., 2018, para referencias que abordan factores de localización). Sin embargo, no parece haber sido abordado en profundidad y/o incluido en el análisis de localización.

La existencia de lealtad espacial implica que los clientes continuarán asignados con su proveedor habitual, independientemente de la localización del competidor más cercano, a menos que la instalación más cercana esté más lejos que cierta distancia admisible, nos referimos a esta distancia como el *radio de lealtad* del cliente $i \in J$. En términos prácticos, nuestro enfoque intenta modelar el comportamiento de un cliente que está dispuesto a caminar una distancia adicional para llegar al restaurante más cercano de su franquicia de comida rápida favorita, independientemente del hecho de que haya una instalación de la competencia abierta más cercana a su lugar de trabajo. Esta idea se resume en la Figura 3.1. Del lado izquierdo muestra la situación de un cliente leal que, aunque se ha cerrado la instalación más cercana de su proveedor preferido (cuadro blanco), permanece leal a la misma empresa (cuadro oscuro) a pesar de que existe una instalación de la competencia (cuadro oscuro) no obstante, hay instalaciones de un competidor (estrella oscura) más cerca a él. Alternativamente, en el lado derecho de la figura se muestra a un cliente no leal al cual no le importará cambiar de proveedor cuando se cierre su instalación preferida.

Por simplicidad, y sin ninguna pérdida de generalidad, asumimos que el radio de lealtad para todos los clientes en el mismo nodo de demanda es idéntico.



(a) Elección de proveedor con lealtad. (b) Elección de proveedor sin lealtad.

Figura 3.1: Elección del proveedor con y sin lealtad.

Para el modelo matemático correspondiente al problema $\overline{(r|p)}^{\mathcal{L}}$- *Centroide* considere los mismos parámetros y variables de decisión que fueron utilizados para modelar el problema $\overline{(r|p)}$-*Centroide* ((3.7) a (3.13)).

PARÁMETROS:

ρ_j**:** Radio de lealtad de los clientes en el nodo de demanda $j \in J$;

CONJUNTOS:

$I^f(j)$**:** Lista ordenada de las instalaciones de la empresa f localizadas dentro del radio de lealtad del cliente $j \in J$, donde $I^f(j) = \left\{i \in I^f : d_{ij} \leq \rho_j\right\}$, para $f \in \{L, F\}$. Los elementos del conjunto $I^f(j)$, satisfacen $d_{I^f(j)_{k-1},j} \leq d_{I^f(j)_k,j}$ para todo $k = 2, \ldots, \left|I^f(j)\right|$;

$I(j)$**:** Lista ordenada de todas las instalaciones localizadas dentro del radio de lealtad

del cliente $j \in J$, donde[1].

$$I(j) = \begin{cases} I^L(j)|I^F(j), & i^*(j) \in I^L \\ I^F(j)|I^L(j), & i^*(j) \in I^F \end{cases},$$

con $i^*(j) = \text{argmin}_{i \in I}\{d_{ij}\}$.

$\overline{I}(j)$**:** Lista ordenada de todas las instalaciones localizadas fuera del radio de lealtad de un cliente, donde $\overline{I}\,(j) = \left\{i \in I^f : d_{ij} > \rho_j\right\}$ y que satisfacen $d_{\overline{I}_{k-1},j} \leq d_{\overline{I}_k,j}$ para $k = 2, \ldots, \left|\overline{I}(j)\right|$, para todo $j \in J$;

$\widetilde{I}^f\,(j)$**:** Lista de las instalaciones de la empresa f ordenadas de acuerdo a su distancia al nodo de demanda j, donde $\widetilde{I}^f\,(j) = I^f\,(j) \cup \left(\overline{I}\,(j) \cap I^f\right), \forall j \in J$ y $f \in \{L, F\}$;

$O(j)$**:** Lista de todas las instalaciones ordenadas según la lealtad y distancia del cliente j, donde $O(j) = \left\{I(j)|\overline{I}(j)\right\}$.

VARIABLES DE DECISIÓN:

z_{ij}**:** Variable binaria que toma valor 1 si el cliente $j \in J$ es atendido por la instalación $i \in I$, y 0 en caso contrario;

FUNCIÓN AUXILIAR:

$P_O\,(S_k)$**:** Función inyectiva que devuelve la posición en la lista ordenada $O\,(j)$ del k-ésimo elemento del conjunto S, donde $P_O\,(S_k) = h \Leftrightarrow O_h = S_k$.

Como en el caso de la sección anterior, el problema $\overline{(r|p)}$-*Centroide* con lealtad de los clientes se puede formular como un problema de programación bi-nivel con la ayuda de un conjunto de restricciones válidas. Esta formulación se presenta en las siguientes líneas:

$$\max_{x^L} \quad \sum_{j \in J} b_j\,(1 - y_j) \tag{3.14}$$

$$s.t. \quad \sum_{i \in I^L} \left(1 - x_i^L\right) = p \tag{3.15}$$

[1]A lo largo de este documento usamos la notación **a**|**b** para representar la concatenación de los vectores **a** y **b**.

$$x_i^L \in \{0,1\}, \qquad \forall i \in I^L \tag{3.16}$$

donde x^F, y y z son soluciones de

$$\text{máx} \quad \sum_{j\in J} b_j y_j \tag{3.17}$$

$$s.t. \quad \sum_{i\in I^F} \left(1 - x_i^F\right) = r \tag{3.18}$$

$$\sum_{i\in I} z_{ij} = 1 \qquad \forall j \in J \tag{3.19}$$

$$z_{ij} \leq x_i^L \qquad \forall i \in I^L, \forall j \in J \tag{3.20}$$

$$z_{ij} \leq x_i^F \qquad \forall i \in I^F, \forall j \in J \tag{3.21}$$

$$\sum_{i\in I^L} z_{ij} \leq 1 - y_j \qquad \forall j \in J \tag{3.22}$$

$$x^L_{\tilde{I}^L(j)_k} + \sum_{\ell=P_O\left(\tilde{I}^L(j)_k\right)+1}^{|O(j)|} z_{O(j)_\ell j} \leq 1 \qquad \forall k = 1,\ldots,\left|\widetilde{I}^L(j)\right|, j \in J, \tag{3.23}$$

$$x^F_{\tilde{I}^F(j)_k} + \sum_{\ell=P_O\left(\tilde{I}^F(j)_k\right)+1}^{|O(j)|} z_{O(j)_\ell j} \leq 1 \qquad \forall k = 1,\ldots,\left|\widetilde{I}^F(j)\right|, j \in J, \tag{3.24}$$

$$x_i^F \in \{0,1\} \qquad \forall i \in I^F \tag{3.25}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{3.26}$$

$$z_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{3.27}$$

Las ecuaciones (3.14)-(3.18), (3.25) y (3.26) tienen la misma interpretación que en la Sección (3.2.1). Las restricciones (3.19) garantizan que cada nodo de demanda sea atendido por una y solo una instalación. Las restricciones (3.20) y (3.21) imponen que un nodo de demanda solo puede ser atendido desde instalaciones que permanecen abiertas, ya sea del líder o del seguidor; las restricciones (3.22) asocian las variables de asignación z y y, lo que obliga a y_j a tomar el valor 1 siempre que el seguidor atienda el nodo de demanda j. Los dos conjuntos de restricciones en (3.23) y (3.24) aseguran que no se asignará un nodo de demanda al seguidor mientras haya una instalación del líder abierta dentro de su radio de lealtad y viceversa. Finalmente, las expresiones en (3.27) son restricciones binarias estándar. Las ecuaciones (3.14)-(3.27) proporcionan una formulación de programación bi-nivel binaria al problema $\overline{(r|p)}^{\mathcal{L}}-$*Centroide*.

Para una solución particular del líder, este problema pueden tener múltiples soluciones óptimas para el seguidor. Si esto ocurre, vamos a adoptar el enfoque optimista para nuestro problema. En nuestro contexto particular, es común considerar

una cooperación entre las empresas que brindar el servicio a los clientes. Esto se puede entender, por ejemplo, si existe un oligopolio de solo dos empresas (es decir, un duopolio) en el cual se considera que ambas hacen un sistema de coalición para emplear prácticas comerciales, por ejemplo inflar los precios en el mercado, restringir la producción, entre otros.

3.2.3 El problema $\overline{(r|p)}^{\mathcal{P}} - Centroide$

Es natural pensar que la lealtad de un cliente a una empresa se puede representar haciendo el uso de preferencias de los clientes. En la literatura podemos encontrar trabajos de programación bi-nivel que han abordado problemáticas que involucran modelos de localización de instalaciones considerando preferencias de los clientes, por ejemplo, en Camacho-Vallejo et al. (2014a), Camacho-Vallejo et al. (2014b), Díaz et al. (2017), Casas-Ramírez et al. (2018), Calvete et al. (2020) por mencionar algunos, quienes para cada cliente dan una lista ordenada de instalaciones de su preferencia para ser atendidos.

En nuestro caso podemos pensar en un ordenamiento asignando un valor de preferencia $P_i^j \in \{1, \ldots, |I|\}$ para cada cliente $j \in J$ hacia cada una de las instalaciones $i \in I$. A la primera, con respecto al ordenamiento de lealtad, se le asigna el valor mayor, es decir $|I|$, a la siguiente en el ordenamiento el valor $|I| - 1$, y así sucesivamente. Tome en cuenta que el ordenamiento que usamos sigue siendo con respecto al radio de lealtad y la instalación a la que está asignado ese cliente.

Ejemplo: Considere una instancia en la cual $|I^L| = |I^F| = 3$, es decir: $|I| = |I^L| + |I^F| = 6$ y se cuenta con 8 clientes $|J| = 8$, si los conjuntos ordenados de lealtad son los siguientes:

$O(1) = \{1, 3, 4, 2, 6, 5\}$

$O(2) = \{1, 2, 5, 6, 3, 4\}$

$O(3) = \{2, 5, 1, 3, 4, 6\}$

$O(4) = \{6, 5, 3, 1, 4, 2\}$

$O(5) = \{4, 3, 2, 6, 1, 5\}$

$O(6) = \{3, 1, 6, 4, 2, 5\}$

$O(7) = \{5, 4, 2, 6, 1, 3\}$

$O(8) = \{6, 1, 2, 4, 3, 5\}$

Podemos definir una matriz de preferencias p_{ij} como sigue:

$$p_{ij} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 6 & 6 & 4 & 3 & 2 & 5 & 2 & 5 \\ 3 & 5 & 6 & 1 & 4 & 2 & 4 & 4 \\ 5 & 2 & 3 & 4 & 5 & 6 & 1 & 2 \\ 4 & 1 & 2 & 2 & 6 & 3 & 5 & 3 \\ 1 & 4 & 5 & 5 & 1 & 1 & 6 & 1 \\ 2 & 3 & 1 & 6 & 3 & 4 & 3 & 6 \end{pmatrix} \end{array}$$

Tomando en cuenta los mismos parámetros, conjuntos y variables de decisión de la subsección anterior donde se definió el problema $\overset{\mathcal{L}}{(r|p)}-Centroide$, podemos definir el siguiente problema equivalente haciendo uso de las preferencias previamente definidas:

$$\max_{x^L} \quad \sum_{j \in J} b_j \left(1 - y_j\right) \tag{3.28}$$

$$\text{Sujeto a:} \quad \sum_{i \in I^L} \left(1 - x_i^L\right) = p \tag{3.29}$$

$$x_i^L \in \{0, 1\}, \qquad \forall i \in I^L \tag{3.30}$$

donde x^F y y son soluciones de

$$\max \quad \sum_{j \in J} b_j y_j \tag{3.31}$$

$$\text{Sujeto a:} \quad \sum_{i \in I^F} \left(1 - x_i^F\right) = r \tag{3.32}$$

$$y_j \geq x_i^F - \sum_{k \in \{I^L : P_O(I_k^L) < P_O(I_i^F)\}} x_k^L \qquad \forall i \in I^F, \forall j \in J \tag{3.33}$$

$$1 - y_j \geq x_i^L - \sum_{k \in \{I^F : P_O(I_k^F) < P_O(I_i^L)\}} x_k^F \qquad \forall i \in I^L, \forall j \in J \tag{3.34}$$

$$x_i^F \in \{0, 1\} \qquad \forall i \in I^F \tag{3.35}$$

$$y_j \in \{0, 1\} \qquad \forall j \in J \tag{3.36}$$

Las ecuaciones (3.28)-(3.30), (3.31), (3.32), (3.35) y (3.36) tienen la misma interpretación que en la Sección (3.2.1). Los dos conjuntos de restricciones en (3.33) y (3.34) aseguran que no se asignará un nodo de demanda al seguidor mientras haya una instalación del líder abierta dentro de su radio de lealtad y viceversa, pero

considerando la preferencia del cliente $j \in J$. Las ecuaciones (3.28)-(3.36) proporcionan una formulación de programación bi-nivel binaria equivalente al problema $\overline{(r|p)}^{\mathcal{L}} - Centroide$ al cual denominaremos $\overline{(r|p)}^{\mathcal{P}} - Centroide$.

Podemos observar que el modelo matemático considerando lealtad es muy diferente al modelo matemático considerando preferencias de los clientes, de entrada vemos que en el primer modelo se cuenta con las restricciones (3.19)-(3.24) y (3.27) las cuales involucran a la variable de asignación z_{ij} mismas que no se tienen en la formulación con preferencias. Es importante notar que la situación que estamos modelando tiene relación con la teoría de juegos o bien los juegos de Stackelberg y que este se considera como un juego de suma cero ya que la demanda total que está en el mercado se asigna completamente al líder o al seguidor al momento de resolver el problema, es decir, si sumamos las funciones objetivos de ambos tomadores de decisiones resulta ser la demanda total de los clientes en el mercado, por ende no es de vital importancia conocer a cual instalación $i \in I$ esta asignado un cliente $j \in J$. Dando así lugar a una formulación más compacta (sin variables de asignación z_{ij}) que ayuda a tener un mejor rendimiento a la hora de resolver las instancias de prueba. Tome en cuenta que ambas formulaciones son completamente diferentes pero equivalentes a la vez, pues las restricciones (3.33)-(3.34) nos ayudan a no violar el criterio asignación con "lealtad" de los clientes a la hora de la resignación a las instalaciones. Además, si por alguna razón los tomadores de decisiones tienen la necesidad de saber cuáles clientes están asignados a las instalaciones abiertas, tome en cuenta que con un breve post-procesamiento una vez teniendo las soluciones óptimas x_i^L, x_i^F del modelo con preferencias y generando los conjuntos ordenados $O(j)$ se puede conocer a cuál instalación se reasignaron los clientes.

Capítulo 4

Métodos exactos

Los métodos exactos son algoritmos que se utilizan para encontrar la solución óptima de un problema de optimización, es decir, la solución que maximiza o minimiza una función objetivo sujeta a un conjunto de restricciones. La principal ventaja estos métodos es que garantizan la obtención de la solución óptima del problema. Sin embargo, su principal desventaja es que pueden requerir un tiempo de ejecución costoso para resolver problemas de gran escala, cosa que no pasa en el problema definido en esta tesis pues no es habitual que una empresa cierre sus instalaciones de un día para otro. Debido a esto, es natural pensar en resolver el problema con alguno de estos métodos, por lo que en esta sección se dará un panorama general sobre los métodos exactos más utilizados en programación bi-nivel y además se describirá un método de enumeración implícita basado en ramificación y corte que se utilizó para resolver el problema bi-nivel $\overline{(r|p)}^{\mathcal{L}}$-*Centroide* así como un algoritmo novedoso propuesto por Fischetti et al. (2017) para resolver problemas de programación bi-nivel enteros mixtos.

4.1 Métodos exactos en programación bi-nivel

Para resolver problemas de programación bi-nivel de forma exacta existen varias técnicas, por mencionar algunas muy conocidas están la de enumeración exhaustiva (también enumeración completa) la cual consiste en enumerar todas las soluciones posibles del líder y para cada una de ellas se enumeran las soluciones del seguidor (o bien se resuelve a optimalidad este nivel con un optimizador comercial) y al final seleccionar la mejor solución bi-nivel según la postura de los tomadores de decisiones, que por lo regular optan por un comportamiento optimista. Sin embargo, debido a la naturaleza exponencial de este enfoque, solo se puede utilizar en instancias pequeñas dependiendo la naturaleza del problema, algunos autores que han utilizado

este enfoque de solución en programación bi-nivel son Dempe and Zemkoho (2011), Chen et al. (2018).

Por otro lado, existen los métodos de descomposición como la descomposición de Benders la cual utiliza la formulación dual del problema bi-nivel llamado problema maestro y lo resuelve de manera iterativa utilizando una técnica de ramificación y corte. En cada iteración, se resuelve un problema denominado subproblema que al resolver obtenemos una cota inferior y por ende un corte que elimina las soluciones no óptimas, este corte se agrega en la siguiente iteración al problema maestro previo y continua el proceso hasta que la solución del problema maestro coincida con la cota, pues se habrá encontrado la solución óptima. Es importante mencionar que si algún problema (maestro o subproblema) es no convexo la descomposición de Benders se convierte en una heurística la cual no asegura encontrar la solución óptima, para tener un panorama más claro de la descomposición de Benders en programación bi-nivel favor de leer Benders (1962), Yuan et al. (2016), Tamura and Nakayama (2018), Shahsavari and Kuhn (2017) y Bard (2013).

Se sabe que en la literatura existen dos técnicas muy utilizadas para resolver problemas combinatorios la ramificación y acotamiento, así como la de ramificación y corte. Estas técnicas son similares en el sentido de que construyen un árbol de búsqueda explícito o implícito, en el que cada nodo representa una solución parcial del problema sin embargo hay una gran diferencia entre ellas por ejemplo en la primera después de cada ramificación, se calcula una cota inferior de la solución en cada subárbol (también llamado nodo) y se elimina aquellos nodos que ya se sabe que no contienen la solución óptima, este proceso se repite hasta que se encuentra la solución óptima o se agotan todas las posibilidades. Por otro lado, en la ramificación y corte, se toma una decisión en cada nodo y se ramifica el árbol en todas las posibles opciones, pero después de cada ramificación, se realiza una verificación de factibilidad para determinar si la solución parcial puede ser mejorada, si se encuentra que la solución parcial no puede ser mejorada, se elimina el nodo correspondiente y se continúa la búsqueda en otro nodo. Estas técnicas son de carácter general y se han adaptado a las necesidades de cada problema a resolver, para tener un panorama general se recomienda leer Bard (2013) además de los siguientes documentos que utilizan dichas técnicas con programación bi-nivel Bard and Moore (1992), Garfinkel and Zhao (2008), Ahmed et al. (2018), entre otros.

4.1.1 Un algoritmo de enumeración implícita de búsqueda en profundidad para el $\overline{(r|p)}^{\mathcal{L}}$-*Centroide*

Como vimos en la sección anterior, es natural pensar resolver un problema de forma exacta, por lo que durante mis estudios de maestría se desarrolló y adapto a las necesidades del problema $\overline{(r|p)}^{\mathcal{L}}$-*Centroide* un algoritmo similar a Bard and Moore (1992) pero tomando en cuenta que hay una gran diferencia en consideraciones importantes como el número de instalaciones a cerrar y la regla de corte.

Para resolver nuestro problema $\overline{(r|p)}^{\mathcal{L}} - Centroide$, consideramos el algoritmo propuesto en Bard and Moore (1992). Ese algoritmo fue propuesto para resolver problemas de programación bi-nivel binarios, por lo que nosotros proponemos una versión adaptada de dicho algoritmo, pero considerando las características de nuestro problema.

La idea es enumerar los nodos de un árbol binario asociado al problema del líder pero teniendo en cuenta la reacción óptima del seguidor. Es decir, el algoritmo encuentra la mejor decisión del líder entre las soluciones factibles de la región inducible IR.

Para iniciar el algoritmo, se busca tener una buena solución del líder. Esto se logra resolviendo de manera iterativa el siguiente problema auxiliar de un solo nivel:

$$\mathcal{Q}(\alpha): \quad \text{máx} \left\{ \sum_{j \in J} b_j y_j : \sum_{j \in J} b_j (1 - y_j) \geq \alpha, (3.15), (3.16), \text{ y } (3.18)\text{-}(3.27) \right\} \tag{4.1}$$

Es fácil ver que solamente agregamos las restricciones del líder al problema del segundo nivel. Además, la función objetivo del líder pasa a ser la restricción $\sum_{j \in J} b_j (1 - y_j) \geq \alpha$, la cual busca establecer una cota inferior para la función objetivo del líder, donde α es un parámetro del modelo que se va determinando conforme avanza el algoritmo.

Antes de describir nuestro algoritmo de solución, necesitamos definir algunos conjuntos y parámetros necesarios para su desarrollo. Además, para mejorar la lectura del mismo, denote una solución del problema bi-nivel como un punto de la forma $XYZ = \{x^L, x^F, y, z\}$ y la función objetivo del líder definida como en (3.14) será denotada por $F = \sum_{j \in J} b_j (1 - y_j)$. Se requiere alguna notación adicional para describir el algoritmo:

PARÁMETROS:

P_k**:** El camino explorado en la iteación k;

$\underline{F}$**:** Cota inferior asociada al valor de la función objetivo del líder. Definida como $\underline{F} = \text{máx}[F : XYZ \in IR^k]$.

NOTACIÓN PARA EL ALGORITMO:

R: Conjunto de radios de lealtad de los clientes, $\mathbf{R} = \{\rho_j, j = 1, \ldots, |J|\}$

B: Conjunto de demandas de los clientes $\mathbf{B} = \{b_j, j = 1, \ldots, |J|\}$

W_k: Conjunto de índices de las variables exploradas durante la k-ésima iteración;

S_k^+: Conjunto de índices de las instalaciones, exploradas durante la k-ésima iteración, que permanecen abiertas después de esa iteración en particular, con $S_k^+ = \left\{i : i \in W_k \wedge x_i^L = 1\right\}$;

S_k^-: Conjunto de índices de las instalaciones, exploradas durante la k-ésima iteración, que se cerraron durante esa iteración en particular, con $S_k^- = \left\{i : i \in W_k \wedge x_i^L = 0\right\}$;

S_k^0: Conjunto de índices de instalaciones no exploradas durante la k-ésima iteración; con $S_1^0 = \{i : i \in W_k\}$;

IR^k: Conjunto de puntos en la región inducible explorada durante la k-ésima iteración.

Es importante mencionar que solamente se va a ramificar en las variables de decisión del líder, esto es, x^L. Note que el algoritmo está diseñado para mejorar la decisión del líder pero considerando la reacción óptima del seguidor, esto es, x^F. Además, dado que la asignación de las variables z y y se hace basada en la lealtad prestablecida de los clientes hacia las instalaciones, estas variables toman menos importancia en el diseño del algoritmo. Al inicio del algoritmo se inicializan los parámetros de la siguiente manera: $\alpha = -\infty$ y $\underline{F} = -\infty$.

El algoritmo propuesto queda como sigue:

Inicialización. Haga $k := 0$, $S_k^+ := \varnothing$, $S_k^- := \varnothing$, $S_k^0 := \{1, \ldots, |I^L|\}$, $\alpha := -\infty$ y $\underline{F} := -\infty$. Esto crea el nodo raíz del árbol de búsqueda.

Paso 1. Iteración general. Haga $x_i^L = 0$ para $i \in S_k^-$ y $x_i^L = 1$ para $i \in S_k^+$. Con el valor actual de α, intente encontrar una solución factible al problema auxiliar $\mathcal{Q}(\alpha)$. Si se encuentra solución, haga $k := k + 1$, guarde la solución XYZ^k y vaya al paso 3; en caso contrario, corte el nodo actual y vaya al paso 6.

Paso 2. Acotamiento. Haga $x^L = (x_i^L)^k$, $\forall i \in I^L$ y resuelva el problema del nivel inferior $\mathcal{L}\left(XYZ^k\right)$ dado por las restricciones $(3.17) - (3.27)$ para obtener una solución $\widehat{XYZ}^k = \left((x^L)^k, (\hat{x}^F)^k, (\hat{y}), (\hat{z})^k\right) \in IR$. Calcule $F(\widehat{XYZ}^k)$ y haga $\underline{F} := \text{máx}[\underline{F}, F(\widehat{XYZ}^k)]$. Si $F(\widehat{XYZ}^k) = F(\widehat{XYZ}^{k-1})$ vaya al paso 6.

Paso 3. Haga $I = \{i \in S_{k-1}^0 : (x_i^L)^k = 0\}$. Si $I = \varnothing$, haga $S_k^+ := S_{k-1}^+$, $S_k^- := S_{k-1}^-$, $S_k^0 := S_{k-1}^0$, $P_k := P_{k-1}$ y vaya al paso 5; en caso contrario, vaya al paso 4.

Paso 4. Ramificación. Crear $|I|$ nuevos nodos como sigue: agregue $i \in I$ al camino P_{k-1} en orden ascendente, uno por uno, para obtener nuevos nodos y el camino actualizado P_k; haga $S_k^+ := S_{k-1}^+$, $S_k^0 := S_{k-1}^0 \backslash I$ y $S_k^- := S_{k-1}^- \cup I$.

Paso 5. Haga $\alpha := \underline{F} + 1$. Regrese al paso 1.

Paso 6. Retroceder. Si no existen nodos sin explorar, vaya al paso 7; en caso contrario, retroceda desde el nodo actual (esto es, al nodo sin explorar agregado más recientemente). Defina la variable correspondiente como i', ramifique sobre su complemento al hacer $(x_{i'}^L)^k = 1$. Actualice los conjuntos S_k^+, S_k^-, S_k^0, y P_k de manera apropiada. Regrese al paso 1.

Paso 7. Terminación. Si $\underline{F} = -\infty$, entonces no hay solución factible para el problema bi-nivel definido por $(3.14) - (3.27)$; en caso contrario, considere el punto factible asociado con $\underline{F}$ como una solución óptima.

A continuación se muestran los Pseudocódigos 1-3 en los cuales se presentan el acotamiento, la ramificación y el retroceso del algoritmo propuesto. En el Pseudocódigo 4, se detalla el algoritmo de enumeración implícita de búsqueda en profundidad, además a manera de ilustrar en la Figura 4.1.2 se muestra el diagrama de flujo del mismo.

Algoritmo 1 ACOTAMIENTO

procedure ACOTAMIENTO(XYZ^k, x^L, I^L)
  Hacer $x^L = (x_i^L)^k, \forall i \in I^L$
  $\widehat{XYZ}^k = \left((x^L)^k, (\hat{x}^F)^k, (\hat{y}), (\hat{z})^k\right) \in IR \leftarrow ResolverNivelInferior(x^L)$
  $F(\widehat{XYZ}^k) \leftarrow Evaluar(\widehat{XYZ}^k)$
  $\underline{F} \leftarrow \text{máx}[\underline{F}, F(\widehat{XYZ}^k)]$
  Return $\widehat{XYZ}^k$, $F(\widehat{XYZ}^k)$, $\underline{F}$
end procedure

Algoritmo 2 RAMIFICACIÓN

procedure RAMIFICACIÓN(I,S_{k-1}^+, $S_{k-1}^0 \backslash I$, $S_{k-1}^- \cup I$)
  Crear $|I|$ nuevos nodos *vivos*: Anexar secuencialmente cada $i \in I$ a P_{k-1} en orden ascendente, para obtener los nuevos nodos *vivos* y el camino final P_k.
  $S_k^+ \leftarrow S_{k-1}^+$
  $S_k^0 \leftarrow S_{k-1}^0 \backslash I$
  $S_k^- \leftarrow S_{k-1}^- \cup I$
  Return S_k^+, S_k^0, S_k^-
end procedure

Algoritmo 3 RETROCESO

procedure RETROCESO(S_k^0)
  if $|S_k^0| = \emptyset$ **then**
    Break
  end if
  Retroceder desde el nodo actual, referirse a la variable asociada como i'.
  Make $(x_{i'}^L)^k = 1$
  Update: S_k^+, S_k^-, S_k^0, y P_k
  Return S_k^+, S_k^-, S_k^0, P_k
end procedure

Algoritmo 4 ALGORITMO DE ENUMERACIÓN IMPLÍCITA DE BÚSQUEDA EN PROFUNDIDAD

procedure AEIBP_Algoritmo(I^L)
  $k \leftarrow 0$
  $S_k^+ \leftarrow \emptyset$
  $S_k^- \leftarrow \emptyset$
  $S_k^0 \leftarrow \{1, \ldots, |I^L|\}$
  $\alpha \leftarrow -\infty$
  $\underline{F} \leftarrow -\infty$
  repeat
    Make $x_i^L = 0 \quad \forall i \in S_k^-$
    Make $x_i^L = 1 \quad \forall i \in S_k^+$
    Resolver el problema $\mathcal{Q}(\alpha)$
    if Factible **then**
      $k \leftarrow k + 1$
      Save XYZ^k
      $\widehat{XYZ}^k, F(\widehat{XYZ}^k), \underline{F} \leftarrow Acotamiento(XYZ^k, x^L, I^L)$
      if $F(\widehat{XYZ}^k) = F(\widehat{XYZ}^{k-1})$ **then**
        $S_k^+, S_k^-, S_k^0, P_k \leftarrow Retroceso(S_k^0)$
      else
        Make $I = \{i \in S_{k-1}^0 : (x_i^L)^k = 0\}$
        if $I = \emptyset$ **then**
          $S_k^+ \leftarrow S_{k-1}^+$
          $S_k^- \leftarrow S_{k-1}^-$
          $S_k^0 \leftarrow S_{k-1}^0$
          $P_k \leftarrow P_{k-1}$
          $\alpha \leftarrow \underline{F} + 1$
        else
          $S_k^+, S_k^0, S_k^- \leftarrow Ramificacion(I, S_{k-1}^+, S_{k-1}^0 \backslash I, S_{k-1}^- \cup I)$
          $\alpha \leftarrow \underline{F} + 1$
        end if
      end if
    else
      $S_k^+, S_k^-, S_k^0, P_k \leftarrow Retroceso(S_k^0)$
    end if
  until $S_k^0 = \emptyset \vee \underline{F} = -\infty$
  if $\underline{F} = -\infty$ **then**
    El problema bi-nivel es infactible
  else
    $\underline{F}$ es el valor óptimo, con $\widehat{XYZ}^k$ asociado.
  end if
  Return $\underline{F}, \widehat{XYZ}^k$
end procedure

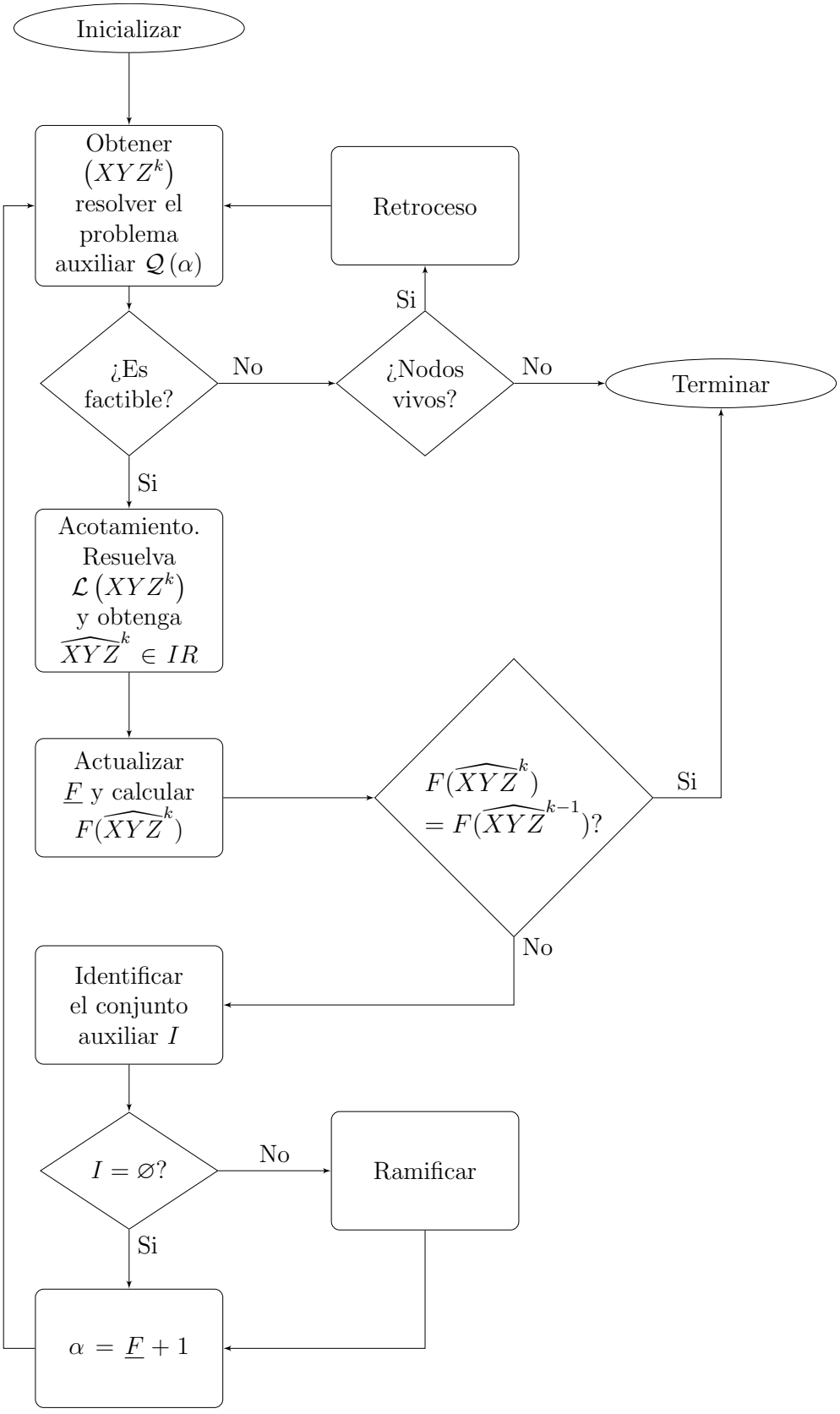


Figura 4.1: Diagrama de flujo del Algoritmo 4

4.1.2 Un algoritmo genérico para resolver problemas bi-nivel entero-mixto

Se sabe que no existe un optimizador comercial para resolver problemas de programación bi-nivel de manera exacta, sin embargo un primer acercamiento a ello es el algoritmo de propósito general para resolver problemas que pertenecen a la programación bi-nivel entera-mixta (MIBLP, pos sus siglas en inglés), el cual se propone en Fischetti et al. (2017), en el cual mediante uso de librerías (basadas en CPLEX), una licencia para utilizarlas así como las instancias del problema bi-nivel en formato MPS (Mathematical Programming System, por sus siglas en inglés) y todo ello montado en el sistema operativo LINUX se puede resolver cualquier problema de esta naturaleza de forma exacta. En el apéndice se muestra a manera de tutorial como es este procedimiento.

El algoritmo en cuestión combina el uso de técnicas de programación lineal, como los planos de corte y la generación de columnas, así como la ramificación y acotamiento para resolver problemas MIBLP. El algoritmo explota la estructura específica de este tipo de problemas para mejorar la eficiencia del proceso de solución. Una contribución importante del algoritmo de Fischetti et al. (2017) es que puede manejar problemas con un gran número de variables enteras, lo que es un desafío significativo para los métodos existentes debido a que esto impacta en el tiempo computacional empleado a la hora de resolver el problema, además la efectividad del algoritmo se demuestra a través de experimentos computacionales en un conjunto de instancias de referencia de la literatura y los resultados muestran que el algoritmo supera a los métodos de otros autores en términos de calidad de solución y tiempo de cálculo.

A continuación mostraremos la idea general de cómo funciona el algoritmo propuesto en Fischetti et al. (2017). Considere el problema de programación bi-nivel entero mixto siguiente:

$$
\begin{aligned}
[MIBLP] \quad & \min_{x,y} \quad c_x^T x + c_y^T y && (4.2)\\
& \text{s.t.} \quad G_x x + G_y y \leq q && (4.3)\\
& \qquad x_j \quad integer, \qquad \forall j \in J_x && (4.4)\\
& \qquad (x, y) \in \mathbb{R}^n && (4.5)\\
& \qquad y \in \arg\min\{d^t y : Ax + By \leq b, \quad y_j \in \mathbb{Z} \quad \forall j \in J_y\} && (4.6)
\end{aligned}
$$

Donde:

- c_x, c_y, q, G_x,G_x, A, B son vectores y matrices de tamaño apropiado.
- $J_x \subseteq N_x$ y $J_y \subseteq N_y$ son los conjuntos de variables de decisión enteras.

Dicho modelo lo podemos escribir de la siguiente manera:

$$\begin{aligned} \min_{x,y} \quad & c_x^T x + c_y^T y && (4.7)\\ \text{s.t.} \quad & G_x x + G_y y \leq q && (4.8)\\ & x_j \in \mathbb{Z}, \qquad \forall j \in J_x && (4.9)\\ & (x, y) \in \mathbb{R}^n && (4.10)\\ & \boldsymbol{d^t y \leq \phi(x)} && (4.11) \end{aligned}$$

Podemos observar que $\phi(x)$ es una función de evaluación por lo cual a la restricción (4.11) se le conoce como la restricción de función de evaluación. Para poder resolver este problema bi-nivel de manera óptima necesitamos conocer $\boldsymbol{d^t y^* \leq \phi(x^*)}$ el cual se obtiene si resolvemos el problema entero mixto siguiente:

$$[MILP] \quad \phi(x^*) = \text{mín} \quad \left\{ d^T y : By \leq b - Ax^*, \quad y_j \in \mathbb{Z} \quad \forall j \in J_y \right\} \qquad (4.12)$$

Debido a que los problemas de programación bi-nivel enteros mixtos pertenecen a la clase de complejidad $\Sigma_2^P - hard$ (demostrado en Lodi et al. (2014)) no hay forma de resolver un problema bi-nivel entero mixto (MIBLP) en tiempo polinomial haciendo uso de un modelo de programación lineal entero mixto MILP de tamaño polinómico a menos que la jerarquía de complejidad colapse (es decir: P=NP), sin embargo, como mencionamos anteriormente, los autores explotan las características del problema y combinan el uso de estas formulaciones con técnicas como los planos de corte, la generación de columnas, así como la ramificación y acotamiento e incluso la ramificación y corte para resolver los problemas bi-nivel de forma exacta haciendo uso de $\phi(x)$. Para ello, veamos la idea general que se presenta en Fischetti et al. (2017) mediante un ejemplo. Primero tome en cuenta la siguiente formulación:

$$\begin{aligned} [MIBLP] \quad \min_{x,y} \quad & c_x^T x + c_y^T y && (4.13)\\ \text{s.t.} \quad & G_x x + G_y y \leq q && (4.14)\\ & Ax + By \leq b && (4.15) \end{aligned}$$

$$(x, y) \in \mathbb{R}^n \tag{4.16}$$
$$\boldsymbol{d^t y \leq \phi(x)} \tag{4.17}$$
$$x_j \in \mathbb{Z}, \qquad \forall j \in J_x \tag{4.18}$$
$$y_j \in \mathbb{Z}, \qquad \forall j \in J_x \tag{4.19}$$

Donde

$$\phi(x) = \text{mín} \quad \left\{ d^T y : By \leq b - Ax, \quad y_j \in \mathbb{Z} \quad \forall j \in J_y \right\} \tag{4.20}$$

Si omitimos la restricción de función de evaluación (4.17) se obtiene un problema MILP al cual los autores llaman problema de relajación de punto alto (High Point Relaxation (HPR)) y definen el problema ($x, y \in \mathbb{R}$) como $\overline{HPR}$. A continuación se presentan ambos modelos:

$$[HPR] \quad \min_{x,y} \quad c_x^T x + c_y^T y \tag{4.21}$$
$$\text{s.t.} \quad G_x x + G_y y \leq q \tag{4.22}$$
$$Ax + By \leq b \tag{4.23}$$
$$(x, y) \in \mathbb{R}^n \tag{4.24}$$
$$x_j \in \mathbb{Z}, \quad \forall j \in J_x \tag{4.25}$$
$$y_j \in \mathbb{Z}, \quad \forall j \in J_x \tag{4.26}$$
$$\tag{4.27}$$

$$\left[\overline{HPR}\right] \quad \min_{x,y} \quad c_x^T x + c_y^T y \tag{4.28}$$
$$\text{s.t.} \quad G_x x + G_y y \leq q \tag{4.29}$$
$$Ax + By \leq b \tag{4.30}$$
$$(x, y) \in \mathbb{R}^n \tag{4.31}$$

Con estos dos modelos como mencionamos anteriormente, los autores explotan sus características para lograr una convergencia a la solución óptima. Por ejemplo, haciendo uso del problema HPR se considera una solución (x, y) no factible bi-nivel si viola la restricción (4.17) y a su vez, una solución $(x, y) \in R^n$, $(n = n1 + n2)$ se considerará factible bi-nivel si satisface todas las restricciones (4.14)–(4.19). Para ello los autores proponen cotas superiores para el seguidor en cada nodo de un árbol de enumeración a las cuales llaman "Follower Upper Bound (FUB) Cuts"para lograr

esto los autores deben utilizar un estimador de $\phi(x)$ lo cual asegura la convergencia al valor óptimo.

A continuación, vamos a describir a grandes rasgos como es el comportamiento de $\phi(x)$ mediante un ejemplo práctico, para ello considere el modelo clásico de programación bi-nivel entero mixto presentado en Bard (2013):

$$\begin{aligned}
&\min_{x} && -x - 10y && (4.32)\\
\text{Sujeto a:}\quad &\min_{y} && y && (4.33)\\
\text{Sujeto a:}\quad & && -25x + 20y \leq 30 && (4.34)\\
& && x + 2y \leq 10 && (4.35)\\
& && 2x - y \leq 15 && (4.36)\\
& && 2x + 10y \geq 15 && (4.37)\\
& && x, y \geq 0, \quad x, y \in \mathbb{Z}_+ && (4.38)
\end{aligned}$$

El problema HPR seria el siguiente:

$$\begin{aligned}
&\min_{x,y\in\mathbb{Z}_+} && -x - 10y && (4.39)\\
\text{Sujeto a:}\quad & && -25x + 20y \leq 30 && (4.40)\\
& && x + 2y \leq 10 && (4.41)\\
& && 2x - y \leq 15 && (4.42)\\
& && 2x + 10y \geq 15 && (4.43)
\end{aligned}$$

En la Figura 4.2 se muestra el espacio de soluciones bi-nivel, mientras que en la Figura 4.3 se muestra la solución óptima de HPR, es decir en $(x^*, y^*) = (2, 4)$.

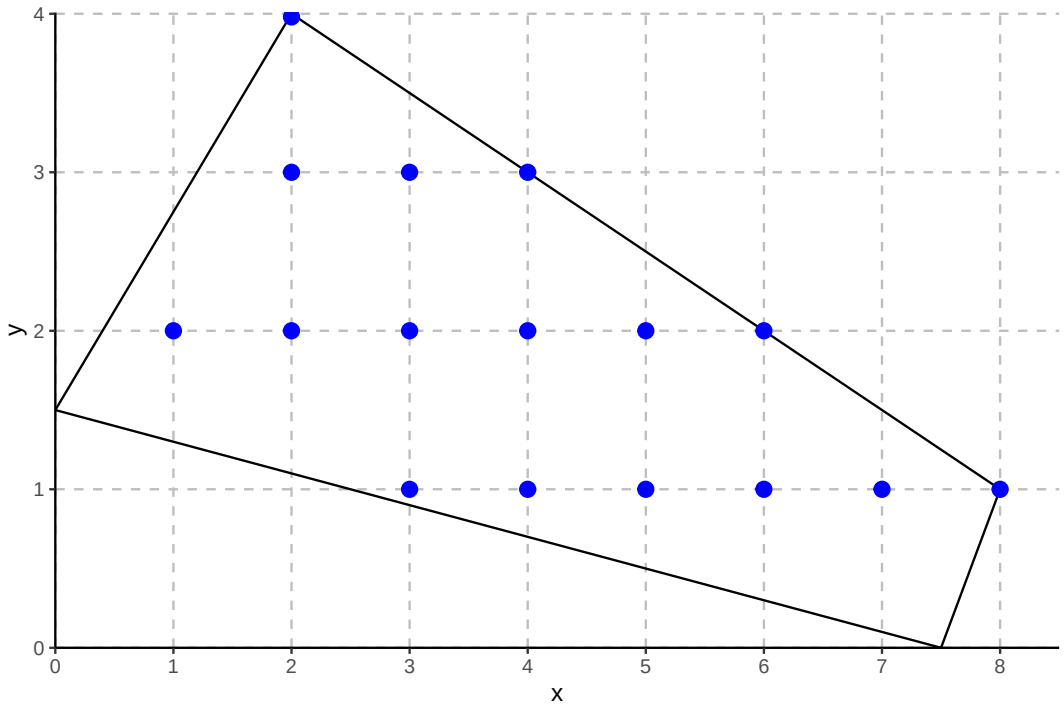


Figura 4.2: Espacio de soluciones bi-nivel Ω.

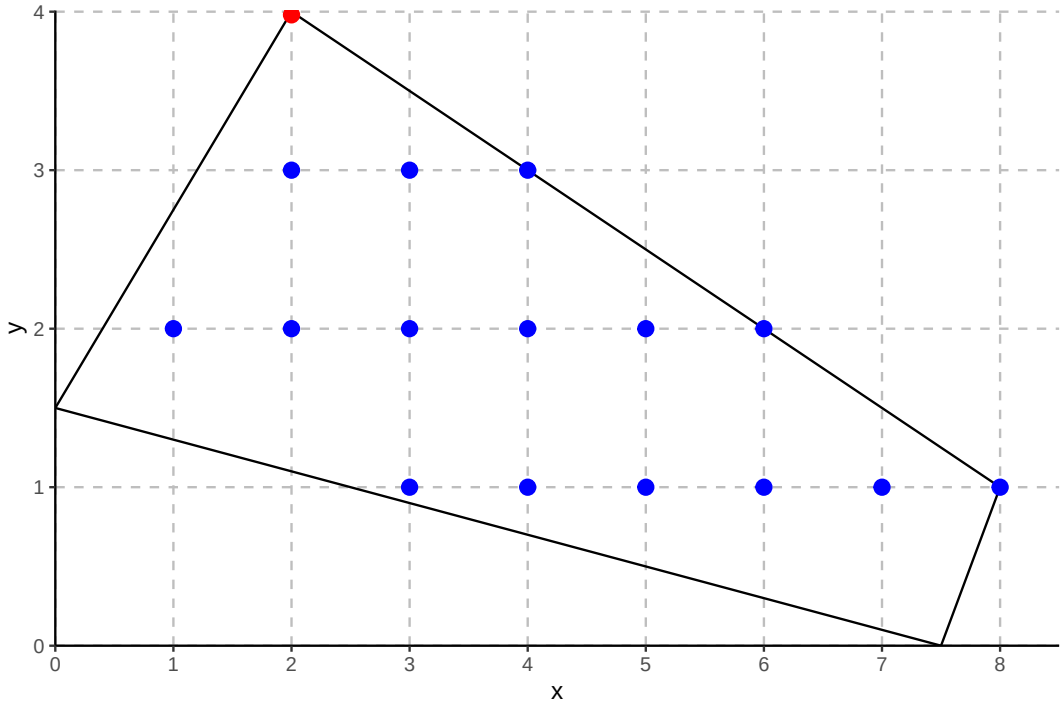


Figura 4.3: Solución de HPR en $(2, 4)$.

En estas formulaciones se omite $\phi(x)$:

$$\begin{aligned}\phi(x) = \text{mín} \quad \{y : & 20y \leq 30 - 25x, \\ & 2y \leq 10 - x, \\ & -y \leq 15 - 2x, \\ & 10y \geq 15 - 2x, \quad y_j \in \mathbb{Z} \qquad \forall j \in J_y\}\end{aligned} \tag{4.44}$$

Tome en cuenta que para problemas de programación bi-nivel lineales, la solución óptima es un vértice del politopo $\overline{HPR} = HPR$, pero para problemas enteros bi-nivel, la solución óptima puede estar en el interior del casco convexo $conv(HPR)$ en este caso sobre $\phi(x)$.

Si resolvemos $\phi(x)$ para $x \geq 0$ y empleando técnicas como lo es la ramificación y acotamiento los autores aseguran encontrar la solución óptima bi-nivel (ver Figura 4.4).

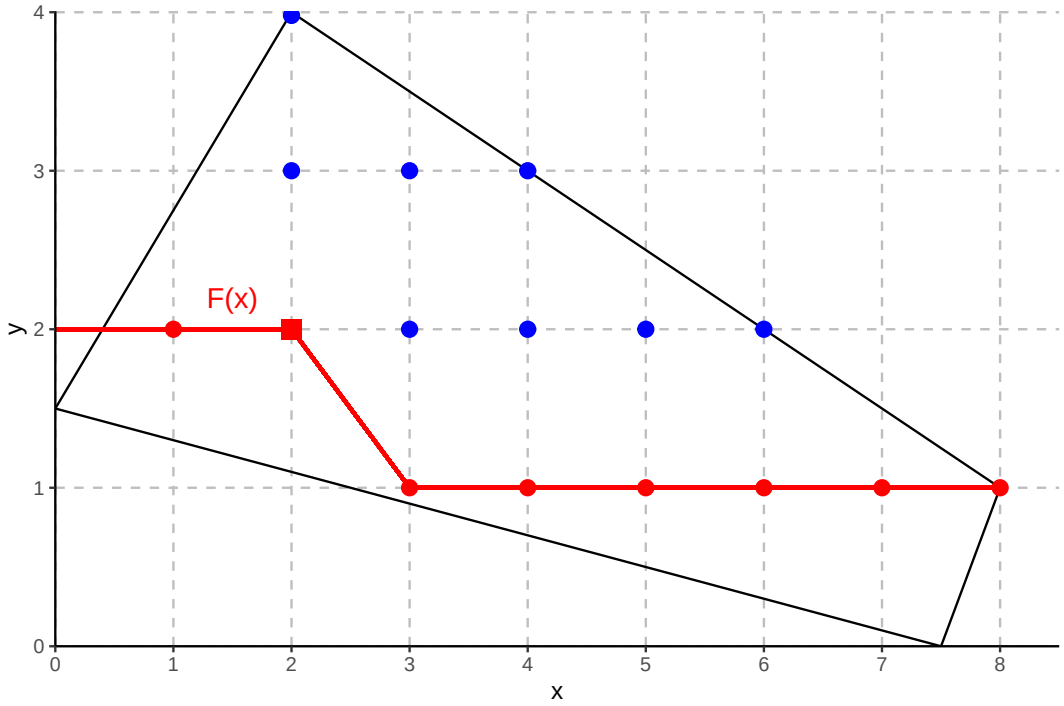


Figura 4.4: La solución óptima bi-nivel pertenece a $\phi(x)$ con $(x^*, y^*) = (2, 2)$.

Además de aplicar la técnica mencionada anteriormente, los autores hacen uso del cono de programación lineal (cono LP), la versión relajada del modelo HPR así

como la técnica de ramificación y corte para agregar planos cortantes (cortes) en la región factible bi-nivel. Por ejemplo, en la Figura 4.5 se muestra en color verde el cono LP que se obtiene al resolver $\overline{HPR}$.

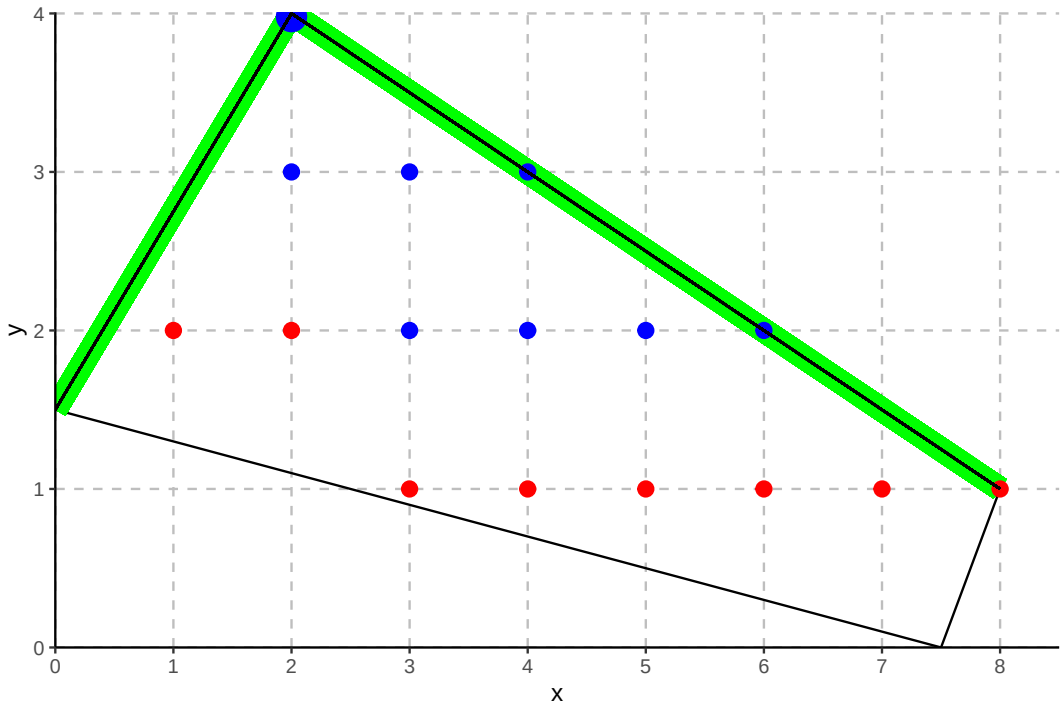


Figura 4.5: Cono LP obtenido resolviendo la relajación de HPR.

Donde podemos apreciar por la ecuación (4.12) que la solución lineal bi-nivel está en $(x^*, y^*) = (2, 4) \rightarrow y^* = 4 > \phi(x^*) = 2$, por lo que $\phi(x^*)$ es el cono en color naranja que a su vez genera el corte $y \leq 2$, dicha información se refleja en las Figuras 4.6 y 4.7 respectivamente.

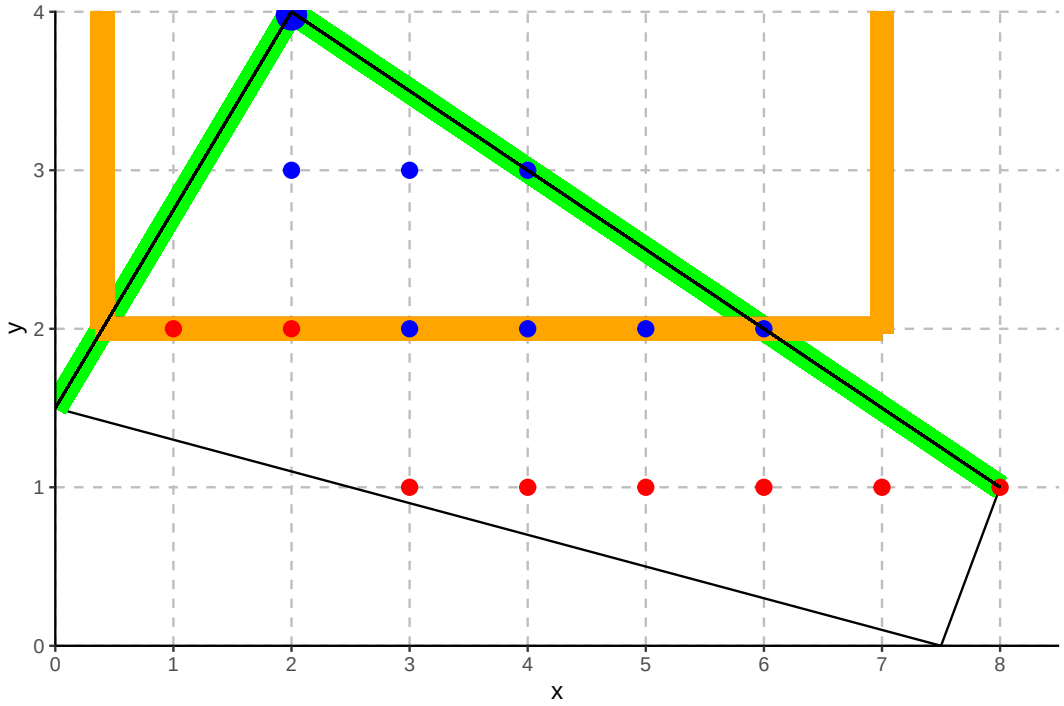


Figura 4.6: Cono generado por $\phi(x^* = 2)$.

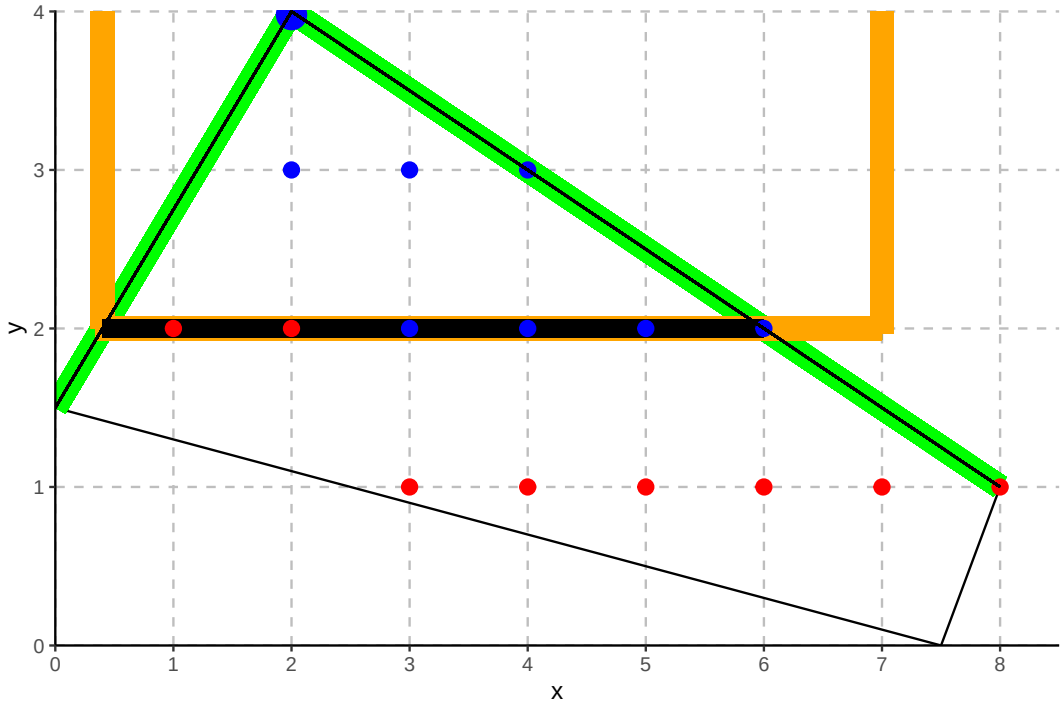


Figura 4.7: Corte en $y \leq 2$.

Actualizando así la región factible bi-nivel (IR) y repitiendo el proceso obtenemos el comportamiento mostrado en las Figuras 4.8 y 4.9.

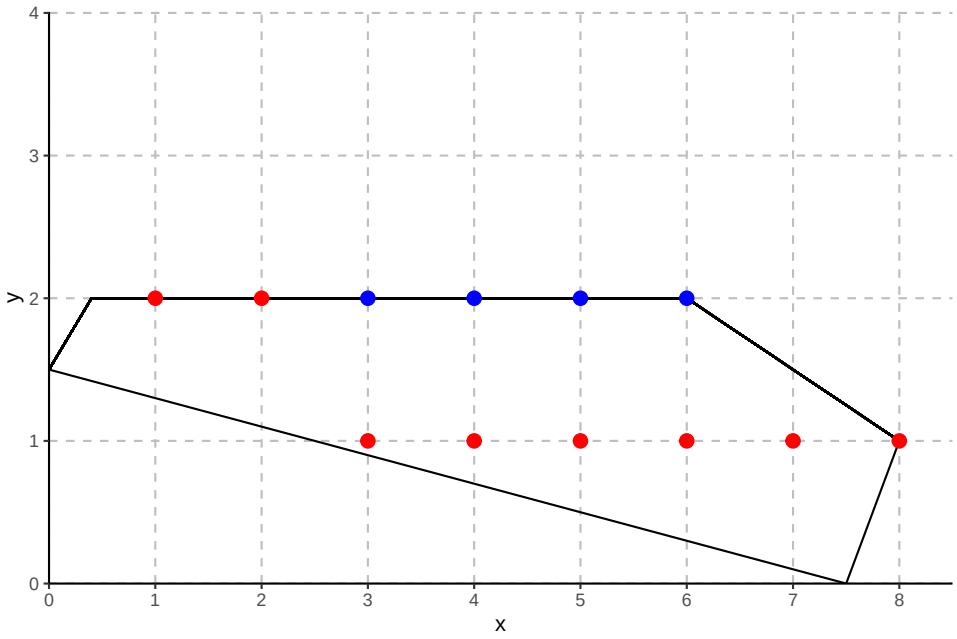


Figura 4.8: Nuevo espacio de soluciones bi-nivel Ω.

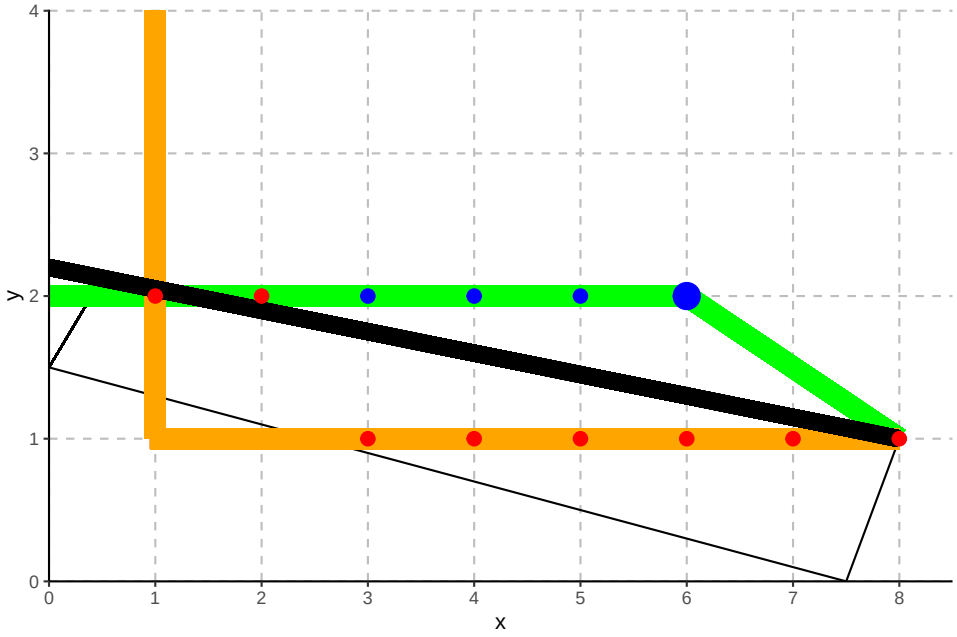


Figura 4.9: Corte generado por $\phi(x^*) = 6$.

En la Figura 4.9 se observa que la solución lineal de $\overline{HPR}$ coincide con la de HPR en $(x^*, y^*) = (2, 2)$. Para profundizar más en el tema se recomienda leer Fischetti et al. (2017, 2018).

Capítulo 5

Metaheuristicas

Por mencionar un poco más de motivación se sabe que los algoritmos de solución se pueden clasificar como métodos exactos o aproximados. Como vimos en la sección anterior los métodos exactos aseguran encontrar la solución óptima en un tiempo razonable, sin embargo, para problemas con clase de complejidad NP-difíciles estos se empiezan a complicar y los métodos exactos podrían necesitar un tiempo de cálculo exponencial en el peor de los casos para resolver una instancia. Esto conduce a menudo a tiempos de cálculo demasiado elevados para fines prácticos. Sin embargo, existen los llamados métodos aproximados como las heurísticas y metaheurísticas las cuales sacrifican la garantía de encontrar soluciones óptimas, pero aseguran obtener buenas soluciones en un tiempo significativamente reducido, por eso es que el uso de este tipo de métodos ha recibido cada vez más atención en los últimos años, los cuales suelen ser más fáciles de implementar que las técnicas clásicas exactas basadas en gradientes. Además, las metaheurísticas no requieren información de gradiente. Esto es conveniente para problemas de optimización donde la función objetivo sólo se da implícitamente (por ejemplo, cuando los valores de la función objetivo se obtienen mediante simulación), o donde la función objetivo no es diferenciable. Es por eso que en esta sección, se presentan dos metaheurísticas, con la finalidad de resolver el problema $\overline{(r|p)}^{\mathcal{L}} - Centroide$, así como el caso donde se modela mediante las preferencias de los clientes. En primer lugar, se introduce una metaheurística conocida como "Greedy Randomized Adaptive Search Procedure" o simplemente GRASP la cual consta de dos fases: una fase constructiva que, como su nombre indica, construye una solución factible y una segunda fase en la que se aplica una búsqueda local a la solución construida con el fin de mejorarla. Después, con el objetivo de intensificar la búsqueda local, se ha implementado una metaheurística híbrida que reemplaza la búsqueda local con otra denominada "Tabu Search" o TS. Este enfoque da lugar un método híbrido que combina ambos procedimientos, a la que denominamos como GRASP-TS.

5.1 GRASP

La metaheurística GRASP fue propuesta por primera vez en Feo and Resende (1995). Es un procedimiento de los denominados multi-arranque que como ya mencionamos es compuesto por dos fases: una constructiva y una de búsqueda local. En cada iteración se construye una nueva solución mediante un procedimiento greedy aleatorio (primera fase) y después a esa solución construida se le aplica una búsqueda local (segunda fase). La solución se construye desde cero, es decir, se selecciona aleatoriamente un elemento de un conjunto potencial de candidatos llamado lista restringida de candidatos *LRC* y este se agrega a la solución que se está construyendo. En terminología del problema bajo estudio esta fase se repite hasta completar una solución factible para el problema del líder (tener p instalaciones cerradas). Una vez construida dicha solución se resuelve el nivel inferior utilizando un optimizador comercial para conocer las funciones objetivo de ambos tomadores de decisiones. Después de ello se realiza una búsqueda local a la solución construida para intentar mejorarla. En el Algoritmo 5 se muestra de manera general el comportamiento del GRASP presentado en Feo and Resende (1995).

Algoritmo 5 GRASP

```
1: InstanciaDeEntrada()
2: repeat
3:     x ← ConstruirSoluciónGreedy()
4:     x' ← BusquedaLocal(x)
5:     Update x ← Mejor(x, x')
6: until Criterio de paro del GRASP no sea satisfecho
7: Return x
```

5.1.1 Descripción del GRASP

Como se acaba de mencionar, el GRASP construye iterativamente una solución factible inicial del líder y posteriormente se busca mejorarla mediante un proceso de búsqueda local el cual explora dentro de un vecindario. La mejor solución encontrada hasta el momento, es decir, la solución incumbente se actualiza siempre que se ha encontrado una mejor solución. El algoritmo termina cuando se cumple un determinado criterio de paro, que en nuestro caso es cuando la función objetivo no ha mejorado en un número predeterminado de iteraciones o bien hasta completar un máximo número de iteraciones.

FASE CONSTRUCTIVA

Como su nombre lo indica, nos enfocaremos en construir una solución inicial para el líder, para lograr este objetivo vamos a crear un conjunto que tenga las p instalaciones a cerrar de tal manera que la perdida de demanda sea la menor posible. Una vez teniendo este conjunto lo traducimos en una solución del líder, para la cual resolvemos el problema del seguidor, obteniendo así una solución factible para el problema bi-nivel. Es natural pensar que podemos elegir p instalaciones a cerrar del conjunto I^L, sin embargo, al hacer esto no estamos dotando de inteligencia y aleatoriedad a la construcción de la solución factible del líder, es por eso que el proceso debe ser de alguna manera inteligente y seleccionar las p instalaciones a cerrar que más le convengan al líder, es decir, cerrar las p con las que la perdida de demanda sea la menor posible. Para lograr crear la solución factible de manera inteligente y a su vez aleatoria necesitamos seleccionar estas p instalaciones del líder del conjunto potencial LRC el cual contiene las mejores opciones para dotar la solución del líder.

Ahora bien, la pregunta es ¿Cómo definimos LRC? para dar respuesta a esto considere una función voraz o también conocida como función greedy que nos ayuda a estimar la pérdida de demanda en la función objetivo del líder para cada instalación candidata a ser cerrada en I^L. Para cada instalación candidata, la función greedy estima la demanda que puede perder el líder si se cierra dicha instalación candidata ya que la demanda capturada por el líder puede mantenerse igual a la captura inicial o puede disminuir debido al cierre de la instalación que se ha seleccionado.

A continuación, explicamos el procedimiento constructivo en detalle. Sea J^L el conjunto de clientes cuya demanda está asignada a una instalación del líder en la configuración inicial (antes del cierre de instalaciones). Por lo tanto, la demanda que puede perder el líder será únicamente la de los clientes que pertenecen al conjunto J^L. Sea también C^L el conjunto de instalaciones que el líder decide cerrar. Sea p_i^j el lugar que ocupa la instalación $i \in I$ en el ordenamiento de lealtad del cliente $j \in J$, de tal manera que la demanda del cliente j se asignará a la instalación abierta con el mínimo valor de p_i^j, es decir, $a_j = \arg\min_{i \in I \setminus C^L}\{p_i^j\}$, donde a_j denota la instalación a la que se asigna la demanda del cliente j.

Sean $L(j)$ y $S(j)$ los conjunto de las instalaciones del líder y del seguidor, respectivamente, dentro del radio de lealtad del cliente j. Asimismo, sean $\overline{L}(j)$ y $\overline{S}(j)$ los mismos conjuntos pero fuera del radio de lealtad.

Para cada $j \in J^L$, el líder puede perder la demanda del cliente j, si está asignado a la instalación $i \in L(j)$ y se satisface alguna de las siguientes condiciones:

1. Si $|L(j) - C^L| = 1$ y $|S(j)| > 0$
2. Si $|L(j) - C^L| = 1$ y $|S(j)| = 0$ y $\text{mín}_{i \in \overline{S}(j)}\{p_i^j\} < \text{mín}_{i \in \overline{L}(j)}\{p_i^j\})$

La primera indica que si dentro del radio de lealtad solo hay una instalación del líder abierta y existen dentro instalaciones del seguidor, este cliente se pierde. La segunda nos dice algo similar pero donde toma en cuenta que dentro del radio de lealtad no hay instalaciones del seguidor, es decir, la única manera de que el cliente se pierda es cuando la instalación más cercana fuera del radio es del seguidor.

Entonces se puede definir la siguiente función auxiliar que evalúa la cantidad de demanda perdida por el líder si se cierra la instalación $i \in I^L$

$$\delta_j(i) = \begin{cases} 1, & \text{Si } a_j = i \text{ y } \Big((|L(j) - C^L| = 1 \text{ y } |S(j)| > 0) \text{ o } (|L(j) - C^L| = 1 \\ & \text{y } |S(j)| = 0 \text{ y } \text{mín}_{i \in \overline{S}(j)}\{p_i^j\} < \text{mín}_{i \in \overline{L}(j)}\{p_i^j\})\Big) \\ 0, & \text{En otro caso.} \end{cases}$$

Se define así la función voraz $g(i)$ para cada instalación $i \in I^L$ del líder como una estimación de la demanda máxima que el líder puede perder si decide cerrar dicha instalación.

$$g(i) = \sum_{j \in J^L : a_j = i} \delta_j(i) b_j$$

Con ella definamos a $g_{\text{mín}} \leftarrow \min_{i \in I^L \setminus C^L}\{g(i)\}$ y $g_{\text{máx}} \leftarrow \max_{i \in I^L \setminus C^L}\{g(i)\}$. Además, sea $\mu = g_{\text{mín}} + \alpha(g_{\text{máx}} - g_{\text{mín}})$ un valor umbral, donde $\alpha \in (0, 1)$ es un parámetro que controla la aleatoriedad del procedimiento greedy. Seleccionamos aleatoriamente una instalación i^* de una lista restringida de candidatos LRC que contiene instalaciones candidatas para las cuales $g(i) \leq \mu$. La instalación i^* se agrega al conjunto de instalaciones cerradas C^L. Al finalizar la iteración se actualizan la función $a(j)$ $\forall j \in J^L$ de manera adecuada, la cual indica la asignación de los clientes. La fase constructiva finaliza cuando se seleccionan p instalaciones a cerrar. Note por ejemplo que si $\alpha = 1$ el valor de $\mu = g_{\text{máx}}$ dando lugar a una selección puramente aleatoria y si $\alpha = 0$ entonces $\mu = g_{\text{mín}}$ dando lugar a una selección puramente miope. El procedimiento constructivo propuesto para el GRASP se muestra en el Algoritmo 6.

Algoritmo 6 Greedy Randomize Procedure

function GreedyRandomizedConstruction(α)
  $C^L := \emptyset$
  $f := 0$
  while ($|C^L| < p$) **do**
    $g(i) := \sum_{j \in J^L : a(j)=i} \delta_j(i) b_j, \forall i \in I^L \setminus C^L$
    $g_{min} := \min_{i \in I^L \setminus C^L} \{g(i)\}$
    $g_{max} := \max_{i \in I^L \setminus C^L} \{g(i)\}$
    $\mu := g_{min} + \alpha(g_{max} - g_{min})$
    $LRC := \{i \in I^L \setminus C^L : g(i) \leq \mu\}$
    Seleccionar $i^\star$ de forma aleatoria de LRC
    $C^L := C^L \cup \{i^\star\}$
    $f := f + g(i)$
    for all ($j \in J^L$) **do**
      if ($a_j = i^\star$) **then**
        $a_j := \arg\min_{i \in I \setminus C^L} \{p_i^j\}$
      end if
    end for
  end while
  Return f
end function

FASE DE BÚSQUEDA LOCAL

En esta segunda fase del GRASP a cada solución obtenida en la fase constructiva se le intenta mejorar mediante una búsqueda local. Con esto se busca intensificar la búsqueda. El procedimiento propuesto explora el vecindario derivado de intercambiar una instalación cerrada por una instalación abierta, es decir, se cierra una instalación abierta $i_1 \in I^L \setminus C^L$ y se abre una instalación cerrada $i_2 \in C^L$. Como se quiere explorar el vecindario para una solución del líder (obtenida de C^L), se denota el vecindario de C^L como $N(C^L)$ y se define como sigue:

$$N(C^L) = \{\overline{C}^L : \overline{C}^L = C^L \setminus \{i_2\} \cup \{i_1\}, i_1 \in I^L \setminus C^L, i_2 \in C^L\}$$

Así, $N(C^L)$ es el conjunto de todas las soluciones que se pueden obtener cerrando una instalación abierta y abriendo una instalación cerrada.

Para cada solución en $N(C^L)$ se resuelve el problema del seguidor para conocer la función objetivo del líder. Además, sea Δ_{i_1,i_2} la función auxiliar que mide el impacto en el valor de la función objetivo del líder; la cual calcula el cambio de demanda de los clientes cuando se cierra la instalación $i_1 \in I^L \setminus x^L$ y se abre una

instalación cerrada $i_2 \in x^L$. Es importante mencionar que en este cambio se considera la reacción del seguidor con respecto a la lealtad de los clientes. Aquí se propone utilizar la estrategia de búsqueda que selecciona la mejor solución del entorno de la solución actual, y el procedimiento termina cuando se obtiene un óptimo local con respecto al entorno $N(x^L)$, es decir, se selecciona la mejor solución en $N(x^L)$ y finaliza cuando el valor de la función objetivo no se puede mejorar más, es decir, cuando todas las soluciones en $N(x^L)$ son peores que la solución incumbente.

En el Algoritmo 7 se muestra el pseudo-código de la fase de búsqueda local. Por lo que el procedimiento general del GRASP propuesto se muestra en el Algoritmo 8. Note que en las líneas siete y nueve se requiere el uso de los Algoritmos 6 y 7, los cuales construyen una solución y se le aplica búsqueda local, respectivamente. También en la línea seis de indica que el método no se detiene hasta llegar a un número máximo de iteraciones o bien tener un número máximo de iteraciones seguidas sin mejora.

Algoritmo 7 Local Search Procedures

1: **function** LocalSearch(C^L)
2: $Best \leftarrow F_{Leader}$
3: Stop $\leftarrow$ **false**
4: **repeat**
5: **for all** $i_2 \in C^L$ and $i_1 \in I^L \setminus C^L$ **do**
6: $\Delta_{i_1,i_2} := SolveLowerLevel(C^L \setminus \{i_2\} \cup \{i_1\})$
7: **end for**
8: $\Delta \leftarrow \text{máx}\{\Delta_{i_1,i_2} : i_2 \in C^L \text{ y } i_1 \in I^L \setminus C^L\}$
9: **if** $\Delta > 0$ **then**
10: $(i_1^*, i_2^*) \in \arg\text{máx}_{i_2 \in C^L, i_1 \in I^L \setminus C^L}\{\Delta_{i_1,i_2}\}$
11: $C^L \leftarrow C^L \cup \{i_1^*\} \setminus \{i_2^*\}$
12: $Best \leftarrow \Delta$
13: **for all** $j \in J^L$ **do**
14: $a(j) \leftarrow \arg\text{mín}_{i \in I \setminus C^L} \{p_i^j\}$
15: **end for**
16: **else**
17: $Stop \leftarrow$ **true**
18: **end if**
19: **until** $Stop$
20: **Return** $Best$
21: **end function**

Algoritmo 8 Heurística GRASP.

```
1: procedure GRASP(MaxIter, α, Increment, MaxStagnant)
2:     IterCount ← 1
3:     MaxStagnant ← 0
4:     F_Leader ← Σ_{j∈J^L} b_j
5:     Best ← F_Leader
6:     while IterCount ≤ MaxIter Or StagnantCount < MaxStagnant do
7:         C^L ← GreedyRandomizedConstruction(α)
8:         F_Leader ← SolveLowerLevel(C^L)
9:         F_Leader ← LocalSearch(C^L)
10:        if F_Leader > Best then
11:            Best ← F_Leader
12:            StagnantCount ← 0
13:        else
14:            StagnantCount ← StagnantCount + 1
15:        end if
16:        IterCount ← IterCount + 1
17:    end while
18: end procedure
```

5.2 Búsqueda Tabú

Por otro lado, el algoritmo de Búsqueda tabú o simplemente TS fue propuesto en Glover (1989, 1990) y se basa en el uso de memoria adaptativa para registrar información histórica del proceso de búsqueda. Además, realiza una exploración inteligente para lograr la efectividad del método. Utiliza estructuras de memoria centrándose en cuatro dimensiones: actualidad, frecuencia, calidad e influencia. Dichas estructuras de memoria ayudan a etiquetar características también conocidos como movimientos en las soluciones previas para mantenerlos prohibidos un número finito de iteraciones, por lo que aquellas soluciones que tengan movimientos prohibidos denominados atributos tabú activos no serán tomadas en cuenta salvo que uno de estos movimientos mejore la solución actual estando prohibido, y así se elimine dicho atributo tabú activo haciendo uso de un criterio de aspiración. Por lo regular, solamente se utilizan estructuras de memoria basadas en la actualidad para evitar ciclos en el método, por lo tanto, el uso de dicha estructura de memoria es una estrategia de exploración agresiva que intenta ir más allá de los óptimos locales, ya que evita volver a explorar las soluciones visitadas en el pasado reciente. En el Algoritmo 9 se muestra de manera general el comportamiento de TS para un problema de minimización donde X es el espacio de soluciones y $N(x)$ denota el vecindario de

la solución $x \in X$.

Algoritmo 9 TS()

```
1: x ← GenerarSolucionInicial()
2: InicializarListasTabú(TL_1, ..., TL_r)
3: while no se cumplan las condiciones de terminación do
4:     N_a(x) ← {x' ∈ N(x) | s' no viola una condición tabú
5:                 o satisface el criterio de aspiración}
6:     x ← argmin{f(x') | x' ∈ N_a(x)}
7:     ActualizarListasTabú(TL_1, ..., TL_r, x, x')
8:     x ← x'
9: end while
```

5.3 Algoritmos híbridos

Como podemos intuir al leer la palabra "híbrido" cuyo significado dice que es la combinación de dos elementos de diferente naturaleza, en el contexto de algoritmos nos referimos a la combinación de dos algoritmos para intensificar la búsqueda de una solución a un problema. Los algoritmos híbridos se han utilizado ampliamente para resolver problemas muy complejos. Se recomienda leer a Blum (2010) quien menciona que podemos distinguir un algoritmo híbrido entre dos categorías: la primera consiste en diseñar un "solucionador" que incluye componentes de una metaheurística en otra (nuestro algoritmo híbrido cae en esta categoría, pues emplea TS en lugar de búsqueda local en el GRASP), mientras que la segunda combina las metaheurísticas con otras técnicas propias de campos como la investigación de operaciones, la inteligencia artificial, entre otras, como es el caso de Lozano and García-Martínez (2010) quien introduce una metaheurística híbrida con algoritmos evolutivos especializados en intensificación y diversificación además de una búsqueda local iterado con una técnica de perturbación evolutiva. Para tener un contexto más amplio sobre metaheurísticas y algoritmos híbridos en general se recomienda leer a Camacho-Vallejo et al. (2023) quienes presentan una amplia revisión de literatura sobre el uso de metaheurísticas a lo largo de la historia desde el punto de vista de programación bi-nivel.

Como ya mencionamos anteriormente estas hibridizaciones potencian las ventajas de utilizar una única metodología para obtener mejores resultados. La combinación de GRASP con TS se estudió por primera vez en Laguna and Velarde (1991). En particular, ya se han propuesto algoritmos GRASP-TS para resolver problemas de manera eficiente (Díaz et al. (2017), Abdinnour-Helm and Hadley (2000), Serra

and Colomé (2001), Delmaire et al. (1999), Díaz et al. (2013), Duarte and Martí (2007) Laguna and Velarde (1991) Lim and Wang (2004)).

El uso de estas metaheurísticas parece ser una buena opción para resolver problemas de localización modelados con programación bi-nivel, en la literatura han hecho uso de ellas por ejemplo en los trabajos de Díaz et al. (2017), Biesinger et al. (2015), Benati and Laporte (1994), Alekseeva et al. (2010). Es por esto que se decidió por diseñar e implementar los algoritmos aquí propuestos para resolver el problema bajo estudio.

A continuación, se describe la metaheurística híbrida que combina el método GRASP descrito en la sección 5.1.1 con TS.

5.3.1 Descripción del GRASP-TS

Como mencionamos al principio de este capítulo con el fin de intensificar la búsqueda combinamos las mataheurísticas GRASP y TS, por lo que en la segunda fase el GRASP propuesto se intercambia la búsqueda local por una búsqueda tabú. Es bien sabido que la búsqueda tabú se usa comúnmente para resolver problemas NP-duros. En este híbrido se utilizan la memoria a corto y largo plazo para escapar de los óptimos locales con el fin de explorar mejor el espacio de soluciones. Se deben definir tres elementos para implementar la búsqueda tabú: atributos tabú, tabú-tenure (un parámetro que indica el número de iteraciones en las que un atributo estará activo) y el criterio de aspiración (criterio para anular los atributos activos tabú). En nuestra implementación, los atributos tabú se definen de manera simple, esto es, una vez que una instalación $i \in C^L$ está abierta, deberá permanecer así durante varias iteraciones.

Para conseguir esto, se mantiene una lista de tamaño fijo con la disciplina de primero en entrar, primero en salir denominada "First-In-First-Out (FIFO, por sus siglas en inglés). Por lo tanto, si la instalación $i^* \in C^L$ se abre en la iteración t, el índice de la instalación i^* se agregará a la lista tabú. Si el tamaño fijo de la lista tabú es ω, el cierre de la instalación i^* estará prohibida en las iteraciones $t+1, t+2, ..., t+\omega$, es importante mencionar que el tamaño de lista tabú es $\omega = |I^L| - p$ para evitar infactibilidad, puesto que debemos asegurar cerrar p instalaciones. También usamos un criterio de aspiración estándar, cada vez que una solución con un atributo tabú activo es mejor que la solución incumbente, se ignoran las restricciones tabú. Por lo tanto, en cualquier iteración del algoritmo, las soluciones candidatas vecinas serán aquellas que no contengan atributos tabú-activos o satisfagan el criterio de aspiración (línea 5 del Algoritmo 10).

Algoritmo 10 Tabu search procedure.

function Tabu Search(C^L)
  $Best \leftarrow F_{Leader}$
  Stop $\leftarrow$ **false**
  repeat
    for all $i_2 \in C^L$ and $i_1 \in I^L \setminus C^L$ and ($i_1 \notin TabuList$ or $\Delta_{i_1,i_2} \geq Best$ **do**
      $\Delta_{i_1,i_2} := SolveLowerLevel(C^L \setminus \{i_2\} \cup \{i_1\})$
    end for
    $\Delta \leftarrow \max\{\Delta_{i_1,i_2} : i_2 \in C^L \text{ y } i_1 \in I^L \setminus C^L\}$
    if $\Delta > 0$ **then**
      $(i_1^*, i_2^*) \in \arg\max_{i_2 \in C^L, i_1 \in I^L \setminus C^L}\{\Delta_{i_1,i_2}\}$
      $C^L \leftarrow C^L \cup \{i_1^*\} \setminus \{i_2^*\}$
      $Best \leftarrow \Delta$
      for all $j \in J^L$ **do**
        $a(j) \leftarrow \arg\min_{i \in I \setminus C^L} \{p_i^j\}$
      end for
    else
      $Stop \leftarrow$ **true**
    end if
  until $Stop$
  Return $Best$
end function

La metaheurística híbrida GRASP-TS se presenta en el Algoritmo 11 donde los criterios de paro son iguales a los de la metaheurística GRASP, es decir al llegar a un número máximo de iteraciones o hasta tener un número máximo de iteraciones seguidas sin mejora.

Algoritmo 11 Heurística GRASP-TS.

```
1: procedure GRASP-TS(MaxIter, α, Increment, MaxStagnant)
2:     IterCount ← 1
3:     MaxStagnant ← 0
4:     F_Leader ← Σ_{j∈J^L} b_j
5:     Best ← F_Leader
6:     while IterCount ≤ MaxIter Or StagnantCount < MaxStagnant do
7:         C^L ← GreedyRandomizedConstruction(α)
8:         F_Leader ← SolveLowerLevel(C^L)
9:         F_Leader ← TabuSearch(C^L)
10:        if F_Leader > Best then
11:            Best ← F_Leader
12:            StagnantCount ← 0
13:        else
14:            StagnantCount ← StagnantCount + 1
15:        end if
16:        IterCount ← IterCount + 1
17:    end while
18: end procedure
```

Capítulo 6

Experimentación Computacional

En esta sección se muestran los resultados numéricos obtenidos al resolver los modelos matemáticos presentados en la Sección 3, mismos que fueron abordados haciendo uso de la lealtad y preferencias de los clientes. Los experimentos se llevaron a cabo utilizando los dos métodos exactos presentados en la Sección 4 así como las dos metaheurísticas descritas en la Sección 5.

6.1 Instancias de prueba

Para evaluar el desempeño de los algoritmos en el modelo $\overline{(r|p)}^{\mathcal{L}} - Centroide$ se utilizaron tres conjuntos de instancias a las cuales denotamos como conjunto A, B Y C, así como también un caso de estudio, mismas que se utilizaron en García-Vélez et al. (2024). En la Tabla 6.1 se muestran los tamaños de cada conjunto de instancias.

Conjunto	Instalaciones	Clientes/nodos de demanda	Instancias	Estrategia
A	10	40	20	Localización aleatoria
B	20	80	11	Localización aleatoria
C	10, 12, 14	100	70	Localización estructurada
Caso Real	20, 21	184	1	Localización estructurada

Tabla 6.1: Características de las instancias de prueba.

Además para evaluar el desempeño de las metaheurísticas se crearon 59 instancias adicionales utilizando el generador ordenado que se describe en García-Vélez (2019) el cual crea un conjunto de N nodos de demanda (clientes) sin repetición en el mallado $[-1, 1] \times [-1, 1]$. Luego, para localizar las p instalaciones existentes del líder se resuelve un problema de la $p - mediana$, lo que permite minimizar la dis-

tancia media entre las instalaciones del líder y los nodos que serán los clientes para después resolver un problema $(r|X_p) - medianoide$ para localizar las r instalaciones del seguidor considerando los $|N| - p$ nodos restantes. Dichos resultados se muestran en esta sección, en las Tablas 6.2 a 6.11.

Para describir el comportamiento de los resultados utilizando el modelo con preferencias, es decir el problema $\overline{(r|p)}^{\mathcal{P}} - Centroide$ solamente se ejecutaron las instancias de los conjuntos A, B y C así como el caso real para los cuales los resultados se muestran en las Tablas 6.12 a 6.17.

6.2 Ambiente computacional

Para llevar a cabo la experimentación computacional se utilizó una estación de trabajo con un procesador Intel (R) Core i7 de 3.6 GHz con 32 GB de memoria RAM y sistema operativo Windows 10 Professional. El método exacto presentado en la Sección 4.1.1 fue implementado en el IDE FICO IVE XPRESS haciendo uso de lenguaje de modelado Mosel 6.2.1, mientras que el método descrito en la Sección 4.1.2 se llevó a cabo utilizando librerías dinámicas de CPLEX montadas en el sistema operativo LINUX utilizando el distro Ubuntu 21.1. Por último, las metaheurísticas se implementaron en el lenguaje de programación $C++$ utilizando Visual Studio 2022 en las cuales para resolver el problema del seguidor se utilizó CPLEX 22.1.1.

Es importante mencionar que las instancias de prueba ejecutadas con las metaheurísticas GRASP y GRASP-TS se ejecutaron diez veces y para darle aleatoriedad a la LRC se usó un valor $\alpha = 0.2$ y en el tamaño de la lista tabú se fijó como el $\omega = \text{mín}\{|I^L| - p, \theta\}$, donde θ es un parámetro para controlar el tamaño de la lista tabú.

6.3 Resultados numéricos utilizando el modelo $\overline{(r|p)}^{\mathcal{L}} - Centroide$

Se muestra de las Tablas 6.2-6.6 el identificador de instancia, $ww.xxx.y - z$, indica el número inicial de instalaciones abiertas (ww); el número de nodos de demanda en la instancia (xxx); el número de instalaciones que se cierran (y); y el identificador de la instancia (z). Asumimos que tanto el número inicial de instalaciones, así como el número de instalaciones a cerrar es el mismo para ambas empresas, es decir $|I^L| = |I^F|$ y $p = r$, respectivamente. En la columna Valor Óptimo como

su nombre lo dice muestra el valor óptimo de esa instancia, es decir el valor óptimo del líder. Las siguientes dos columnas muestran el tiempo de ejecución en segundos para los métodos exactos descritos en la Sección 4 mismos que se definen en García-Vélez et al. (2024) y en Fischetti et al. (2017) respectivamente. Luego de ello, se presentan los resultados obtenidos al utilizar las metaheurísticas, con el fin de identificar que tan alejados están los valores incumbentes del óptimo denotado por $F^L_{Óptimo}$, en las tablas se presentan las holguras de optimalidad ($\%GAP$) para el mejor valor encontrado (BV), el valor promedio de las diez corridas (AV) y el peor valor encontrado (WV), para cada metaheurística, denotado por $F_{Líder}$. Dicha holgura se calcula como sigue: $GAP = \left|\frac{F^L_{Óptimo} - F_{Líder}}{F^L_{Óptimo}}\right| \times 100\,\%$. Además, se presentan los tiempos promedio de ejecución para ambas metaheurísticas, el cual se calcula como sigue: $Tiempo = \frac{\sum_{i=1}^{10} TiempoEjecucion_i}{10}$, recuerde que cada instancia se ejecutó diez veces por metaheurístico, por lo que $TiempoEjecucion_i$ indica el tiempo de ejecución de una instancia en la corrida $i = 1, 2, \ldots, 10$. Por último, se muestra el ahorro en tiempo de ejecución al utilizar el GRASP-TS sobre el GRASP el cual se calcula de la siguiente manera: $Ahorro = \left|\frac{Tiempo_{\text{GRASP}} - Tiempo_{\text{GRASP-TS}}}{Tiempo_{\text{GRASP}}}\right| \times 100\,\%$.

Para describir los resultados de cada tabla nos referiremos como "primer método" al algoritmo exacto propuesto en García-Vélez et al. (2024) y a su vez nos referiremos como "segundo método" al propuesto por Fischetti et al. (2017).

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP BV	% GAP GRASP AV	% GAP GRASP WV	% GAP GRASP-TS BV	% GAP GRASP-TS AV	% GAP GRASP-TS WV	Tiempo promedio (seg.) GRASP	Tiempo promedio (seg.) GRASP-TS	Ahorro (%)
10.40.2–1	1,510	2.2	0.9	–	–	–	–	–	–	0.1	0.0	46.3
10.40.2–2	1,863	2.8	0.4	–	–	–	–	–	–	0.2	0.1	14.8
10.40.2–3	1,078	2.1	0.4	–	–	–	–	–	–	0.1	0.0	53.9
10.40.2–4	1,629	1.8	0.4	–	–	–	–	–	–	0.1	0.1	22.4
10.40.2–5	1,969	1.9	0.3	–	–	–	–	–	–	0.2	0.1	71.7
10.40.3–1	1,510	2.0	0.7	–	–	–	–	–	–	0.1	0.1	45.0
10.40.3–2	1,962	9.0	0.7	–	–	–	–	–	–	1.2	0.3	78.7
10.40.3–3	1,078	8.7	0.7	–	–	–	–	–	–	0.4	0.2	61.2
10.40.3–4	1,629	6.9	0.7	–	–	–	–	–	–	0.1	0.0	57.5
10.40.3–5	1,927	9.2	0.7	–	–	–	–	–	–	58.0	0.2	99.6
10.40.4–1	1,520	8.0	1.3	–	–	–	–	–	–	10.5	0.5	94.9
10.40.4–2	2,053	22.0	1.1	–	–	–	–	–	–	1.8	0.3	86.2
10.40.4–3	1,197	23.1	1.3	–	–	–	–	–	–	0.6	0.3	51.8
10.40.4–4	1,629	18.5	1.1	–	–	–	–	–	–	10.4	0.4	96.6
10.40.4–5	1,852	23.3	1.2	–	–	–	–	–	–	5.4	0.4	93.4
10.40.5–1	1,520	32.1	1.2	–	–	–	–	–	–	0.5	0.6	−35.7
10.40.5–2	2,122	29.5	1.4	–	–	–	–	–	–	0.1	0.2	−135.6
10.40.5–3	1,316	38.8	1.2	–	–	–	–	–	–	1.7	0.4	74.2
10.40.5–4	1,672	32.9	1.2	–	–	–	–	–	–	2.9	0.5	82.8
10.40.5–5	1,843	33.5	1.9	–	–	–	–	–	–	1.2	0.3	71.9

Tabla 6.2: Instancias del conjunto A.

En la Tabla 6.2 se muestran las instancias del conjunto A, podemos observar que en cuanto a tiempo de ejecución tuvo muy buen comportamiento el segundo método a comparación del primero pues en todas las instancias este resuelve en menos de 2 segundos pues la instancia más tardada es la $10.40.5 - 5$ mientras que

el primero tarda a lo menos 1.8 segundos como se puede observar en la instancia $10.40.2 - 4$. En cuanto a las metaheurísticas se aprecia que en todas las instancias ambas logran encontrar el valor óptimo pues el $\%GAP$ se muestra cómo "-" y el tiempo de ejecución en promedio no rebasa los 10.5 segundos para el GRASP y para el GRASP-TS no rebasa los 0.6 segundos, debido a ello se tiene un ahorro en tiempo considerable teniendo ahorros desde un 14.8 % hasta un 96.6 %, solamente las instancias $10.40.5 - 1$ y $10.40.5 - 2$ de las 20 instancias de este conjunto tardan en promedio más tiempo con el metaheurístico GRASP-TS.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP BV	AV	WV	% GAP GRASP-TS BV	AV	WV	Tiempo promedio (seg.) GRASP	GRASP-TS	Ahorro (%)
20.80.3–1	4,282	144.6	28.0	–	–	–	–	–	–	1.4	0.3	76.2
20.80.3–2	3,360	145.5	29.0	–	–	–	–	–	–	0.9	0.3	71.0
20.80.3–3	3,985	77.1	26.9	–	–	–	–	–	–	0.3	0.3	19.4
20.80.3–4	3,921	143.6	27.4	–	–	–	–	–	–	0.8	0.2	78.4
20.80.3–5	4,087	122.0	25.5	–	–	–	–	–	–	1.0	0.6	44.3
20.80.5–1	4,282	3,135.8	413.3	–	–	–	–	–	–	6.9	1.6	76.3
20.80.5–2	3,360	2,187.1	409.0	–	–	–	–	–	–	1.6	0.4	74.8
20.80.5–3	3,985	2,408.5	396.5	–	–	–	–	–	–	11.0	1.3	88.4
20.80.5–4	3,921	3,085.4	398.3	–	–	–	–	–	–	1.8	0.7	63.7
20.80.5–5	4,087	3,082.3	379.9	–	–	–	–	–	–	11.3	1.9	83.5
20.80.8–1	4,222	47,488.0	3,000.8	–	–	–	–	–	–	112.1	4.8	95.7

Tabla 6.3: Instancias del conjunto B.

Los resultados del conjunto B de instancias se muestran en la Tabla 6.3, en la cual se observa que el tiempo de ejecución con el segundo método está muy por debajo a comparación de los tiempos que tardo en resolver el primero, por ejemplo en la instancia $20.80.8 - 1$ tarda $47,488$ segundos (aprox. 13.2 horas) con el primer método mientras que el segundo tardo solo 3000.8 segundos (aprox. 50 minutos) aun y cuando los tiempos de ejecución del primer método son mucho mayores que el segundo. Para el caso de las metaheurísticas se observa que en todas las ejecuciones se logra encontrar el valor óptimo además el tiempo computacional de ejecución es menor con GRASP-TS a comparación del GRASP en todas las instancias. Para la mayoría de instancias con $p = r = 3, 5$ se obtiene un ahorro entre el 70 % y 90 % a su vez el ahorro más bajo es de 19.4 % para la instancia $20.80.3 - 3$ y únicamente la instancia $20.80.8 - 1$ es la que presenta el mejor ahorro con 95.7 % algo positivo debido a la combinatoria que requiere resolver esta instancia.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP			% GAP GRASP-TS			Tiempo promedio (seg.)		Ahorro (%)
				BV	AV	WV	BV	AV	WV	GRASP	GRASP-TS	
10.100.2–1	4,485	6.2	0.9	–	–	–	–	–	–	0.4	0.2	51.6
10.100.2–2	4,595	6.7	0.8	–	–	–	–	–	–	0.6	0.3	54.9
10.100.2–3	4,477	6.8	0.9	–	–	–	–	–	–	0.9	0.5	47.2
10.100.2–4	5,298	5.4	0.8	–	–	–	–	–	–	0.9	0.3	68.8
10.100.2–5	3,974	5.8	1.1	–	–	–	–	–	–	1.2	0.4	70.4
10.100.3–1	4,481	29.2	2.4	–	–	–	–	–	–	1.5	0.6	60.6
10.100.3–2	4,604	27.4	1.7	–	–	–	–	–	–	0.1	0.4	−159.3
10.100.3–3	4,500	25.0	1.9	–	–	–	–	–	–	2.4	0.5	77.0
10.100.3–4	5,389	23.2	2.1	–	–	–	–	–	–	1.0	0.4	62.4
10.100.3–5	3,998	26.0	1.8	–	–	–	–	–	–	0.7	0.6	11.4
10.100.4–1	4,345	68.6	3.6	–	–	–	–	–	–	6.3	0.6	90.1
10.100.4–2	4,604	62.1	3.9	–	–	–	–	–	–	5.0	3.7	25.5
10.100.4–3	4,463	58.1	2.9	–	–	–	–	–	–	1.8	0.9	50.9
10.100.4–4	5,158	53.5	4.4	–	–	–	–	–	–	0.6	0.5	11.3
10.100.4–5	4,028	54.4	3.8	–	–	–	–	–	–	0.7	0.8	−10.5
10.100.5–1	4,193	93.6	7.6	–	–	–	–	–	–	3.5	1.1	68.7
10.100.5–2	4,607	88.8	6.8	0.04	0.04	0.04	–	–	–	183.6	1.6	99.1
10.100.5–3	4,237	98.5	3.8	–	–	–	–	–	–	1.5	1.0	30.4
10.100.5–4	4,799	84.7	4.7	–	–	–	–	–	–	0.8	0.7	6.8
10.100.5–5	3,874	89.2	6.2	–	–	–	–	–	–	21.8	8.8	59.6

Tabla 6.4: Instancias del conjunto C con 10 instalaciones.

Con respecto al conjunto C de instancias, en particular, cuando se tienen 10 instalaciones abiertas por competidor, los resultados se muestran en la Tabla 6.4. Para los métodos exactos se observa que el segundo método no excede los 7.6 segundos como podemos observar para la instancia $10.100.5-1$ mientras que el primero se observa que a lo menos tarda 5.4 segundos en resolver, como el caso de la instancia $10.100.2-4$. En el caso de las metaheurísticas se puede observar que el GRASP-TS alcanza el valor óptimo en todas las corridas para todas las instancias, pero el GRASP no lo logra en la instancia $10.100.5-2$ la cual presenta el tiempo de ejecución más alto con 183.6 segundos, esto debido a la localización de las instalaciones y el numero $p = r = 5$ de instalaciones a cerrar la cual representa el 50 % de las instalaciones que operan con normalidad, sin embargo, la holgura de optimalidad es muy baja con un $\%GAP = 0.04$. Por otro lado, con respecto al tiempo computacional requerido, el GRASP-TS es más rápido que el GRASP en la mayoría de instancias, solamente en las instancias $10.100.3-2$ y $10.100.4-5$ el GRASP-TS tardó más que el GRASP, esto se ve reflejado en el ahorro negativo en dichas instancias. En la mayoría de las instancias se muestra un buen ahorro, solo en las instancias $10.100.2-3$, $10.100.3-5$, $10.100.4-2$, $10.100.4-4$, $10.100.5-3$ y $10.100.5-4$ se obtuvo un ahorro promedio menor al 50 %, siendo esta última la de menor ahorro con 6.8 %. En muchas de las instancias se presenta un ahorro promedio entre el 50 % y 90 % mientras que solo las instancias $10.100.4-1$ y $10.100.5-2$ son las que presentan el mayor ahorro, pues en ellas se reduce el tiempo en más del 90 %, sin embargo, es claro que en esta última instancia fue debido al tiempo que tardo en resolverse con GRASP, pues resulto ser la más tardada.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP			% GAP GRASP-TS			Tiempo promedio (seg.)		Ahorro (%)
				BV	AV	WV	BV	AV	WV	GRASP	GRASP-TS	
12.100.2–1	4,561	10.3	1.7	–	–	–	–	–	–	1.8	0.4	75.4
12.100.2–2	4,448	10.2	1.3	–	–	–	–	–	–	174.6	0.5	99.7
12.100.2–3	4,271	10.5	1.4	–	–	–	–	–	–	1.5	0.6	62.6
12.100.2–4	5,030	10.2	1.4	–	–	–	–	–	–	0.5	0.1	74.9
12.100.2–5	4,058	9.8	1.3	–	–	–	–	–	–	0.6	0.4	27.0
12.100.3–1	4,553	55.0	3.8	–	–	–	–	–	–	3.1	0.6	81.9
12.100.3–2	4,425	48.5	3.3	–	–	–	–	–	–	0.8	0.8	−3.4
12.100.3–3	4,408	54.4	3.6	–	–	–	–	–	–	4.8	0.7	85.0
12.100.3–4	5,192	47.0	3.6	–	–	–	–	–	–	3.3	0.8	74.6
12.100.3–5	3,926	46.3	4.8	–	–	–	–	–	–	2.1	0.8	62.5
12.100.4–1	4,510	144.1	9.8	–	–	–	–	–	–	3.8	0.8	78.3
12.100.4–2	4,418	137.6	9.2	–	–	–	–	–	–	3.1	0.9	69.2
12.100.4–3	4,329	147.6	7.5	–	–	–	–	–	–	2.3	0.9	59.9
12.100.4–4	5,299	144.6	7.4	–	–	–	–	–	–	4.6	1.1	76.8
12.100.4–5	3,819	135.1	7.3	–	–	–	–	–	–	10.0	1.2	88.3
12.100.5–1	4,358	289.8	13.9	–	–	–	–	–	–	1.0	0.4	62.9
12.100.5–2	4,378	286.6	13.7	–	–	–	–	–	–	1.5	1.0	32.7
12.100.5–3	4,161	273.4	10.1	–	–	–	–	–	–	11.5	2.5	78.5
12.100.5–4	5,334	280.3	11.5	–	–	–	–	–	–	15.1	1.5	89.7
12.100.5–5	3,709	273.5	14.4	0.01	0.01	0.01	–	–	–	314.7	3.3	99.0
12.100.6–1	4,257	439.0	18.2	–	–	–	–	–	–	0.8	0.7	9.7
12.100.6–2	4,409	398.0	14.0	–	–	–	–	–	–	4.9	1.0	80.0
12.100.6–3	4,203	287.2	14.7	–	–	–	–	–	–	46.8	2.2	95.3
12.100.6–4	5,348	371.4	13.4	–	–	–	–	–	–	16.8	1.6	90.8
12.100.6–5	3,664	370.0	18.3	–	–	–	–	–	–	57.1	1.9	96.6

Tabla 6.5: Instancias del conjunto C con 12 instalaciones.

En la Tabla 6.5 se muestran los resultados de las instancias del conjunto C pero con 12 instalaciones abiertas por competidor en la cual se observa que el primer método tarda más tiempo en encontrar la solución óptima pues los tiempos de ejecución oscilan entre los 9.8 y 439 segundos como podemos observar en las instancias $12.100.2-5$ y $12.100.6-1$ respectivamente, mientras que para el segundo método estos oscilan entre 1.3 y 18.3 segundos como vemos en las instancias 12.100.2-2 y $12.100.6-1$. En cuanto a las metaheurísticas de nueva cuenta el GRASP-TS alcanza el valor óptimo en todas las corridas para todas las instancias, mientras que el GRASP no lo logra en la instancia $12.100.5-5$. Con respecto al tiempo computacional requerido, el GRASP-TS es más rápido que el GRASP en la mayoría de instancias, salvo para la instancia $12.100.3-2$. En la mayoría de las instancias se muestra un buen ahorro, solo en las instancias $12.100.2-5$, $12.100.5-2$ y $12.100.6-1$ se obtuvo ahorro promedio menor al 35 %. De manera similar que en la Tabla 6.4, hay muchas instancias que reportan un ahorro promedio entre el 50 % y 90 %. Las instancias $12.100.2-2$, $12.100.5-5$, $12.100.6-3$, $12.100.6-4$ y $12.100.6-5$ son las que presentan el mayor ahorro, pues en todas ellas se reduce el tiempo en más del 90 %.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP BV	AV	WV	% GAP GRASP-TS BV	AV	WV	Tiempo promedio (seg.) GRASP	GRASP-TS	Ahorro (%)
14.100.2–1	4,801	20.7	2.2	–	–	–	–	–	–	1.3	0.8	35.2
14.100.2–2	4,830	17.0	2.1	–	–	–	–	–	–	0.9	0.2	76.5
14.100.2–3	4,070	18.9	2.1	–	–	–	–	–	–	3.0	0.8	73.1
14.100.2–4	5,002	12.7	2.9	–	–	–	–	–	–	4.7	0.8	83.9
14.100.2–5	4,267	17.2	2.0	–	–	–	–	–	–	3.3	0.7	79.4
14.100.3–1	4,700	103.1	7.6	–	–	–	–	–	–	9.8	1.7	83.0
14.100.3–2	5,000	73.2	6.6	–	–	–	–	–	–	4.1	0.9	77.8
14.100.3–3	4,043	103.8	7.2	–	–	–	–	–	–	1.0	1.0	4.9
14.100.3–4	4,972	87.6	6.9	–	–	–	–	–	–	1.6	1.2	22.4
14.100.3–5	4,162	91.5	6.6	–	–	–	–	–	–	2.5	1.4	44.9
14.100.4–1	4,597	333.5	17.4	–	–	–	–	–	–	21.3	2.1	90.2
14.100.4–2	5,036	309.5	16.4	–	–	–	–	–	–	9.6	1.1	88.9
14.100.4–3	4,059	336.2	17.2	–	–	–	–	–	–	24.1	1.7	93.0
14.100.4–4	4,876	314.9	19.1	–	–	–	–	–	–	8.6	1.9	77.9
14.100.4–5	4,168	310.3	19.3	–	–	–	–	–	–	6.3	1.7	72.9
14.100.5–1	4,491	748.9	42.5	–	–	–	–	–	–	41.7	3.6	91.4
14.100.5–2	5,103	684.7	37.3	–	–	–	–	–	–	4.0	1.2	71.6
14.100.5–3	4,034	749.4	43.1	–	–	–	–	–	–	6.1	2.0	67.7
14.100.5–4	4,871	687.2	39.3	–	–	–	–	–	–	8.3	2.0	75.5
14.100.5–5	4,208	714.5	41.8	–	–	–	–	–	–	14.5	1.9	86.9
14.100.6–1	4,418	1,249.7	59.0	–	–	–	–	–	–	14.5	2.6	82.2
14.100.6–2	5,159	1,206.7	54.6	–	–	–	–	–	–	7.8	1.6	79.9
14.100.6–3	3,993	1,248.4	51.8	–	–	–	–	–	–	40.2	6.0	85.2
14.100.6–4	4,892	1,193.4	64.4	–	–	–	–	–	–	10.5	2.2	79.2
14.100.6–5	4,108	1,206.2	62.3	–	–	–	–	–	–	13.1	3.2	75.5

Tabla 6.6: Instancias del conjunto C con 14 instalaciones.

En la Tabla 6.6 se muestran los resultados del conjunto de instancias C con 14 instalaciones abiertas por competidor. Al igual que las tablas de resultados anteriores el segundo método resuelve en menor tiempo a comparación del primero ya que tiene tiempos de ejecución que van desde los 2 a 64.4 segundos mientras que el primer método va de 12.7 a 1249.7 segundos. En los resultados de las metaheurísticas vemos que ambos algoritmos encuentran el valor óptimo en todas las ejecuciones además resulta que el GRASP-TS es más rápido que el GRASP en todos los casos. En la mayoría de las instancias se muestra un buen ahorro, solo en las instancias $14.100.2-1$, $14.100.3-3$, $14.100.3-4$, $14.100.3-5$ y $14.100.3-5$ se obtuvo ahorro promedio menor al $45\,\%$. Sin embargo, en la mayoría de las instancias se tiene un ahorro promedio entre el $50\,\%$ y $90\,\%$. Las instancias $14.100.4-1$, $14.100.4-3$ y $14.100.5-1$ son las que presentan el mayor ahorro.

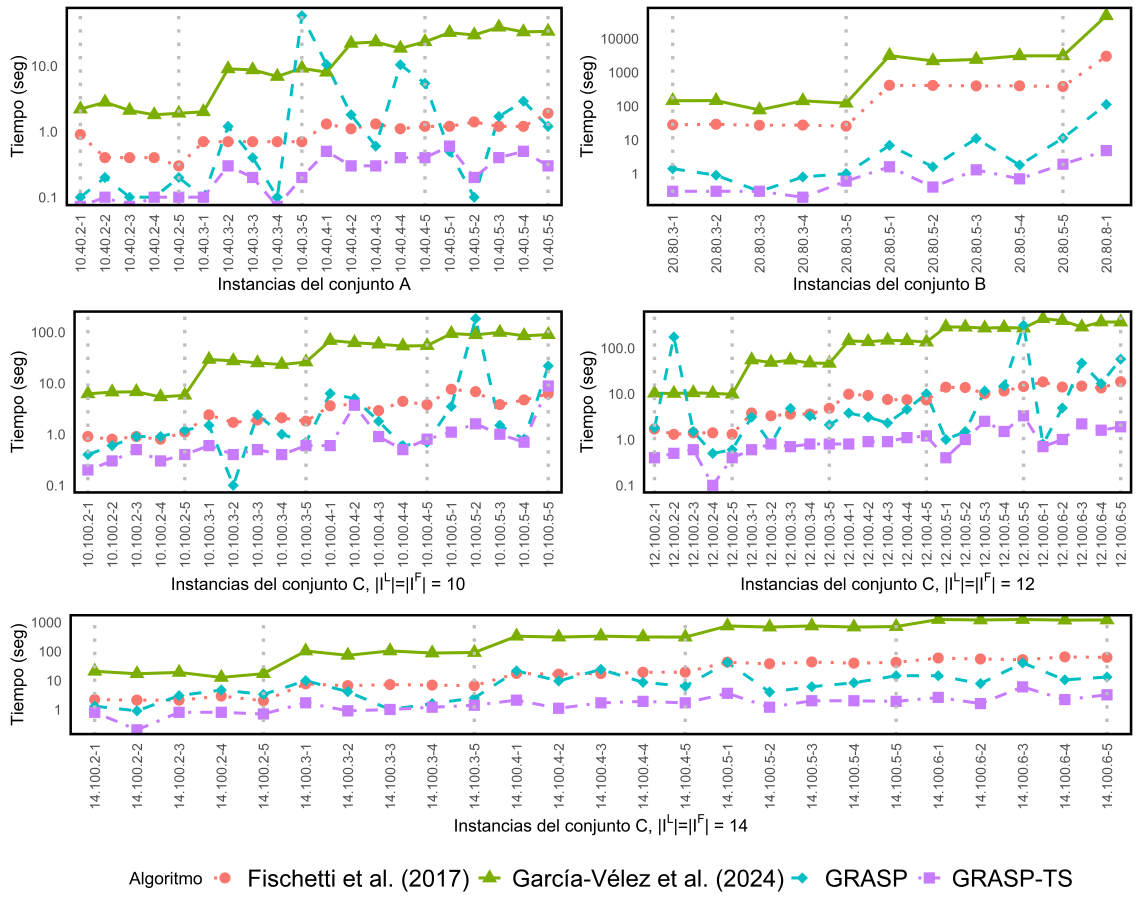


Figura 6.1: Ilustración del tiempo computacional requerido por cada algoritmo probado usando el modelo $\overline{(r|p)}^{\mathcal{L}} - Centroide$.

En la Figura 6.1 se muestra el rendimiento de los algoritmos probados usando el modelo donde se considera la lealtad de los clientes, en términos del tiempo computacional requerido, se traza el tiempo de ejecución para cada instancia. En general, se observa que GRASP-TS es el algoritmo que requiere menos tiempo para resolver el problema.

Instancia p=r	Valor Óptimo	García-Vélez (2019) Tiempo(seg.)	% GAP GRASP BV	% GAP GRASP AV	% GAP GRASP WV	% GAP GRASP-TS BV	% GAP GRASP-TS AV	% GAP GRASP-TS WV	Tiempo promedio (seg.) GRASP	Tiempo promedio (seg.) GRASP-TS	Ahorro (%)
2	64,613	43.9	–	–	–	–	–	–	0.7	0.1	79.9
3	64,613	233.1	–	–	–	–	–	–	3.7	0.9	75.3
4	64,613	1,485.8	–	–	–	–	–	–	4.6	1.6	64.9
5	64,613	7,031.9	–	–	–	–	–	–	11.8	1.1	90.6
6	64,613	22,077.3	–	–	–	–	–	–	12.4	5.4	56.4
7	64,613	51,925.8	–	–	–	–	–	–	76.0	9.3	87.8

Tabla 6.7: Caso real para diferentes valores de $p = r$.

Por último, en la Tabla 6.7 se muestran los resultados del caso de estudio donde se consideran 184 clientes con 20 y 21 instalaciones para el líder y seguidor, respectivamente. Debido a que ya conocemos el comportamiento tendencial en todos los casos (Tablas 6.2-6.6) donde vemos que el segundo método es más rápido en tiempo que el primero, para estas instancias se decidió solamente ejecutar con el primer método. Se hicieron pruebas con diferentes valores de p y r y vemos que en todos los casos el valor óptimo encontrado es $64,613$ y a mayor número de $p = r$ el tiempo se incrementa, el caso más tardado es cuando $p = r = 7$ donde el tiempo de ejecución fue de 51,925.8 segundos o bien aproximadamente 14.4 horas similar al tiempo que tardo la instancia $20.80.8 - 1$ de la Tabla 6.3. El comportamiento de ambas metaheurísticas se mantiene, es decir, llegan al óptimo en todas las corridas y el GRASP-TS consume menor tiempo computacional. En la mayoría de los casos se muestra un buen ahorro, solo cuando $p = r = 6$ el ahorro promedio es menor a 60 %. Para los casos donde $p = r \in \{2, 3, 4, 7\}$ se presenta un ahorro promedio entre el 60 % y 90 %. Solo para el caso donde $p = r = 5$ el tiempo se reduce en más del 90 %.

En las Tablas 6.8-6.11 se muestran los resultados de la experimentación computacional adicional. Instancias de mayor tamaño fueron generadas, para las cuales se desconoce el valor óptimo. Es por esto, que en las primeras dos columnas se muestra el mejor valor encontrado (incumbente) por GRASP y GRASP-TS, respectivamente. La holgura de optimalidad es calculada en base a este valor.

Instancia	Incumbente		% GAP GRASP			% GAP GRASP-TS			Tiempo promedio (seg.)		Ahorro
	GRASP	GRASP-TS	BV	AV	WV	BV	AV	WV	GRASP	GRASP-TS	(%)
14.250.2–1	11,050	11,050	–	–	–	–	–	–	5.9	3.2	45.6
14.250.2–2	10,848	10,848	–	–	–	–	–	–	5.5	2.0	63.0
14.250.2–3	10,195	10,195	–	–	–	–	–	–	5.2	2.2	58.0
14.250.2–4	10,365	10,365	–	–	–	–	–	–	7.2	3.3	53.7
14.250.2–5	9,929	9,929	–	–	–	–	–	–	30.2	1.4	95.3
14.250.3–1	10,935	10,935	–	–	–	–	–	–	10.8	4.8	55.7
14.250.3–2	10,877	10,877	–	–	–	–	–	–	9.4	3.3	65.4
14.250.3–3	10,140	10,140	–	–	–	–	–	–	12.9	3.0	76.9
14.250.3–4	10,361	10,361	–	–	–	–	–	–	14.6	3.5	75.9
14.250.3–5	9,671	9,671	–	–	–	–	–	–	2.7	2.3	14.8
14.250.4–1	10,808	10,808	–	–	–	–	–	–	17.5	4.0	77.2
14.250.4–2	10,924	10,924	–	–	–	–	–	–	20.9	5.5	73.8
14.250.4–3	10,159	10,159	–	0.1	1.2	–	–	–	28.0	4.4	84.2
14.250.4–4	10,380	10,380	–	–	–	–	–	–	25.7	4.8	81.5
14.250.4–5	9,713	9,713	–	–	–	–	–	–	15.6	3.3	78.6
14.250.5–1	10,779	10,779	–	–	–	–	–	–	50.4	6.4	87.3
14.250.5–2	10,948	10,948	–	–	–	–	–	–	27.3	4.3	84.2
14.250.5–3	10,312	10,312	–	0.3	1.2	–	–	–	144.3	7.5	94.8
14.250.5–4	10,458	10,458	–	–	–	–	–	–	66.5	5.8	91.3
14.250.5–5	9,578	9,578	–	–	–	–	–	–	11.3	4.3	62.4
14.250.6–1	10,898	10,898	–	–	–	–	–	–	38.2	6.4	83.3
14.250.6–2	10,851	10,851	–	–	–	–	–	–	31.4	5.5	82.7

Tabla 6.8: Instancias con 14 instalaciones y 250 clientes.

En la Tabla 6.8 se puede observar que en todas las instancias el valor incumbente coincide en ambas metaheurísticas, solamente las instancias con identificador de instancia tres para $p = r \in \{4, 5\}$ este valor no se encuentra en las diez corridas en el caso del GRASP mientras que en el GRASP-TS logra encontrar el mismo incumbente en el total de sus corridas. Con respecto al tiempo computacional requerido, el GRASP-TS es más rápido que el GRASP en todos los casos pues el ahorro siempre es positivo, en la mayoría de las instancias se muestra un buen ahorro promedio entre el 50 % y 90 %. Hay tres instancias que presentan el mayor ahorro, ya que se reduce el tiempo en más del 90 %.

Instancia	Incumbente		% GAP GRASP			% GAP GRASP-TS			Tiempo promedio (seg.)		Ahorro
	GRASP	GRASP-TS	BV	AV	WV	BV	AV	WV	GRASP	GRASP-TS	(%)
9.150.2	62,231	62,231	–	–	–	–	–	–	11.4	0.6	94.7
9.150.4	62,455	62,455	–	–	–	–	–	–	25.9	1.5	94.1
12.150.2	61,098	61,098	–	–	–	–	–	–	2.6	0.9	64.9
12.150.4	60,888	60,888	–	–	–	–	–	–	57.5	2.6	95.4
12.150.6	59,486	59,486	–	–	–	–	–	–	281.8	10.8	96.2
15.150.2	62,566	62,566	–	–	–	–	–	–	49.1	2.0	95.9
15.150.4	63,867	63,867	–	–	–	–	–	–	107.9	4.4	95.9
15.150.6	62,398	62,315	–	–	–	–	–	–	187.7	8.3	95.6
20.150.2	61,500	61,500	–	–	–	–	–	–	3.8	1.9	50.3
20.150.4	60,581	60,157	–	–	–	–	–	–	231.3	6.7	97.1
20.150.6	59,379	59,069	–	–	–	–	–	–	433.1	13.1	97.0
30.150.2	64,413	64,413	–	–	–	–	–	–	10.3	0.8	91.8
30.150.4	64,413	64,413	–	–	–	–	–	–	49.8	6.9	86.1
40.150.2	66,717	66,717	–	–	–	–	–	–	2.5	1.0	60.9
40.150.3	66,717	66,717	–	–	–	–	–	–	5.7	1.2	78.5
50.150.2	71,466	71,466	–	–	–	–	–	–	8.4	0.4	95.3
50.150.3	71,466	71,466	–	–	–	–	–	–	54.1	0.4	99.2

Tabla 6.9: Instancia Red de Swain extendida con 150 clientes.

Los resultados del segundo conjunto de instancias grandes se muestran en la Tabla 6.9 donde podemos observar que en todas las instancias el incumbente coincide para ambos metaheurísticos, además que este fue encontrado en las diez corridas para ambos casos. En cuanto a tiempo de ejecución resulta que el GRASP-TS es más rápido que el GRASP en todos los casos. Cabe destacar que en 12 de las 17 instancias se obtiene una reducción en el tiempo mayor al 90 %. No está de más mencionar que estas instancias se obtuvieron a partir de la Red de Swain desarrollada por Ruiz-Hernández et al. (2017) mismas que se usaron en la tesis García-Vélez (2019).

Instancia	Incumbente		% GAP GRASP			% GAP GRASP-TS			Tiempo promedio (seg.)		Ahorro
	GRASP	GRASP-TS	BV	AV	WV	BV	AV	WV	GRASP	GRASP-TS	(%)
20.250.8–1	10,912	10,912	–	–	–	–	–	–	1,186.3	197.9	83.3
20.250.8–2	10,817	10,817	–	–	–	–	–	–	940.0	162.3	82.7
20.250.8–3	10,927	10,927	–	0.0	0.2	–	–	–	6,060.3	1,233.8	79.6
20.250.8–4	10,722	10,722	–	–	–	–	–	–	1,322.0	287.8	78.2
20.250.8–5	11,234	11,234	–	–	–	–	–	–	876.3	150.1	82.9
20.250.10–1	11,301	11,301	–	–	–	–	–	–	1,423.6	432.5	69.6
20.250.10–2	10,906	10,906	–	–	–	–	–	–	2,638.3	696.4	73.6
20.250.10–3	10,821	10,821	–	–	–	–	–	–	1,798.0	1,557.0	13.4
20.250.10–4	10,776	10,776	–	–	–	–	–	–	2,853.1	1,100.1	61.4
20.250.10–5	11,070	11,070	–	–	–	–	–	–	986.7	194.0	80.3

Tabla 6.10: Instancias de 20 instalaciones y 250 clientes.

En la Tabla 6.10 se puede observar que en todas las instancias el incumbente coincide de nueva cuenta para ambos algoritmos, solamente la instancia $20.250.8-3$ resuelta con GRASP no encuentra dicho valor en el total de sus corridas mientras que el GRASP-TS si lo consigue. En cuanto a tiempo computacional el GRASP-TS resuelve más rápido que el GRASP, por lo que en la mayoría de las instancias se muestra un buen ahorro, todas ellas por encima del 60 % salvo la instancia $20.250.10-3$ la cual presenta el menor ahorro con 13.4 %. Cabe mencionar que para este conjunto de instancias, en ninguna se reduce el tiempo a más del 84 %.

Instancia	Incumbente		% GAP GRASP			% GAP GRASP-TS			Tiempo promedio (seg.)		Ahorro
	GRASP	GRASP-TS	BV	AV	WV	BV	AV	WV	GRASP	GRASP-TS	(%)
50.500.7–1	25,658	25,658	–	–	–	–	–	–	18,885.3	2,863.0	84.8
50.500.7–2	24,573	24,573	–	–	–	–	–	–	19,736.5	2,790.9	85.9
50.500.7–3	24,529	24,529	–	–	–	–	–	–	20,600.4	3,009.6	85.4
50.500.7–4	25,989	25,989	–	–	–	–	–	–	24,408.2	3,402.7	86.1
50.500.7–5	26,265	26,265	–	–	–	–	–	–	24,034.4	3,353.9	86.1
75.500.7–1	25,825	25,825	–	–	–	–	–	–	39,132.8	5,884.4	85.0
75.500.7–2	24,634	24,634	–	–	–	–	–	–	43,848.0	6,005.5	86.3
75.500.7–3	26,335	26,335	–	–	–	–	–	–	64,923.4	9,894.0	84.8
75.500.7–4	26,715	26,715	–	–	–	–	–	–	49,871.8	8,554.2	82.9
75.500.7–5	25,299	25,299	–	–	–	–	–	–	54,629.6	8,536.0	84.4

Tabla 6.11: Instancias de 50 y 75 instalaciones y 500 clientes.

Por último, en la Tabla 6.11 se puede observar el mismo comportamiento eficiente de ambos metaheuristicos, ambos logran encontrar el mismo valor incumbente en todas sus corridas. El tiempo computacional requerido para resolver las instancias es más favorable para el GRASP-TS pues en todos los casos tarda mucho menos que el GRASP. Algo importante de resaltar es que para este conjunto de instancias, el tiempo computacional se reduce más del 80 % pero menos del 90 %.

6.4 Resultados numéricos utilizando el modelo $\overline{(r|p)}^{\mathcal{P}} - Centroide$

En esta sección mostramos los resultados de las Tablas 6.2-6.7 pero haciendo uso del modelo con preferencias de los clientes en lugar de lealtad. Por lo que en las Tablas 6.12-6.17 se muestra el identificador de instancia, el valor óptimo y el tiempo de ejecución de ambos métodos exactos, los resultados de ambas metaheurísticas con la holgura de optimalidad correspondiente así como su ahorro promedio de igual forma que como se abordó en la Sección 6.3. Es importante hacer notar que si aparece "< 0.0" en las tablas de resultados, indica que el algoritmo requirió menos de 0.009 segundos para ejecutarse.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP			% GAP GRASP-TS			Tiempo promedio (seg.)		Ahorro (%)
				BV	AV	WV	BV	AV	WV	GRASP	GRASP-TS	
10.40.2–1	1,510	1.5	0.2	–	–	–	–	–	–	<0.0	<0.0	0.0
10.40.2–2	1,863	1.4	0.2	–	–	–	–	–	–	<0.0	<0.0	0.0
10.40.2–3	1,078	1.4	0.2	–	–	–	–	–	–	<0.0	<0.0	0.0
10.40.2–4	1,629	1.1	0.2	–	–	–	–	–	–	<0.0	<0.0	0.0
10.40.2–5	1,969	1.2	0.2	–	–	–	–	–	–	<0.0	<0.0	0.0
10.40.3–1	1,510	4.2	0.3	–	–	–	–	–	–	<0.0	<0.0	0.0
10.40.3–2	1,962	5.3	0.3	–	–	–	–	–	–	0.3	0.1	66.0
10.40.3–3	1,078	4.0	0.4	–	–	–	–	–	–	<0.0	<0.0	0.0
10.40.3–4	1,629	3.0	0.4	–	–	–	–	–	–	<0.0	<0.0	0.0
10.40.3–5	1,927	5.0	0.3	–	–	–	–	–	–	3.5	0.1	97.0
10.40.4–1	1,520	10.6	0.5	–	–	–	–	–	–	0.1	0.2	−100.0
10.40.4–2	2,053	12.4	0.5	–	–	–	–	–	–	0.5	0.1	80.0
10.40.4–3	1,197	12.7	0.5	–	–	–	–	–	–	0.2	0.1	50.0
10.40.4–4	1,629	10.1	0.6	–	–	–	–	–	–	0.9	0.3	66.0
10.40.4–5	1,852	10.8	0.4	–	–	–	–	–	–	0.1	0.2	−100.0
10.40.5–1	1,520	16.1	0.6	–	–	–	–	–	–	0.1	0.2	−100.0
10.40.5–2	2,122	16.1	0.5	–	–	–	–	–	–	0.3	0.2	33.0
10.40.5–3	1,316	18.8	0.6	–	–	–	–	–	–	0.2	0.2	0.0
10.40.5–4	1,672	16.8	0.6	–	–	–	–	–	–	0.2	0.3	−50.0
10.40.5–5	1,843	15.7	0.7	–	–	–	–	–	–	0.3	0.2	50.0

Tabla 6.12: Instancias del conjunto A.

En la Tabla 6.12 se observar que en cuanto a tiempo de ejecución tuvo muy buen comportamiento el segundo método a comparación del primero pues en todas las instancias este resuelve en menos de 0.7 segundos pues la instancia más tardada es la $10.40.5-5$ mientras que el primero tarda a lo menos 1.1 segundos como se puede observar en la instancia $10.40.2-4$. Se aprecia que en todas las instancias ambas metaheurísticas logran encontrar el valor óptimo en el total de las diez corridas. En cuanto a tiempo de ejecución en promedio se observa que el GRASP en casi todas sus instancias resuelve en menos de un segundo, salvo la instancia $10.40.3-5$ la cual tarda 3.5 segundos mientras que para el GRASP-TS no rebasa los 0.3 segundos. En promedio, GRASP-TS reduce el tiempo del GRASP en un 3.85 %.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP BV	AV	WV	% GAP GRASP-TS BV	AV	WV	Tiempo promedio (seg.) GRASP	GRASP-TS	Ahorro (%)
20.80.3–1	4,282	60.5	12.1	–	–	–	–	–	–	0.2	0.2	0.0
20.80.3–2	3,360	36.4	8.6	–	–	–	–	–	–	0.2	0.1	−50.0
20.80.3–3	3,985	25.6	8.1	–	–	–	–	–	–	<0.0	0.1	−200.0
20.80.3–4	3,921	49.0	11.9	–	–	–	–	–	–	0.3	<0.0	500.0
20.80.3–5	4,087	43.3	10.7	–	–	–	–	–	–	0.1	0.1	0.0
20.80.5–1	4,282	1,408.5	91.1	–	–	–	–	–	–	0.2	0.4	−100.0
20.80.5–2	3,360	625.4	81.2	–	–	–	–	–	–	0.1	0.1	0.0
20.80.5–3	3,985	945.8	76.6	–	–	–	–	–	–	0.2	0.5	−150.0
20.80.5–4	3,921	1,412.7	114.8	–	–	–	–	–	–	0.2	<0.0	300.0
20.80.5–5	4,087	1,301.2	76.5	–	–	–	–	–	–	0.3	0.6	−100.0
20.80.8–1	4,222	28,346.0	814.6	–	–	–	–	–	–	10.2	2.7	74.0

Tabla 6.13: Instancias del conjunto B.

En la Tabla 6.13, se presentan las instancias del conjunto B. Se puede observar que, en términos de tiempo de ejecución, una vez más, el segundo método es más rápido que el primero en todas las instancias. También se nota que, una vez más, ambas metaheurísticas alcanzan el valor óptimo en todas las instancias durante las diez ejecuciones. La instancia $20.80.8 - 1$ tarda más tiempo en promedio, siendo resuelta en 10.2 segundos y 2.7 segundos para cada metaheurística, respectivamente. Similarmente, se obtiene una pequeña reducción en el tiempo promedio con GRASP, en general para las 11 instancias en el conjunto B.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP BV	AV	WV	% GAP GRASP-TS BV	AV	WV	Tiempo promedio (seg.) GRASP	GRASP-TS	Ahorro (%)
10.100.2–1	4,485	2.8	0.6	–	–	–	–	–	–	<0.0	<0.0	0.0
10.100.2–2	4,595	2.7	0.5	–	–	–	–	–	–	<0.0	<0.0	0.0
10.100.2–3	4,477	3.1	0.4	–	–	–	–	–	–	0.2	0.1	50.0
10.100.2–4	5,298	2.9	0.5	–	–	–	–	–	–	0.2	0.1	50.0
10.100.2–5	3,974	2.6	0.5	–	–	–	–	–	–	0.1	0.1	0.0
10.100.3–1	4,481	14.3	0.7	–	–	–	–	–	–	0.1	0.1	0.0
10.100.3–2	4,604	10.8	0.7	–	–	–	–	–	–	0.3	0.2	33.0
10.100.3–3	4,500	13.4	0.7	–	–	–	–	–	–	0.3	0.3	0.0
10.100.3–4	5,389	11.4	0.8	–	–	–	–	–	–	0.1	0.1	0.0
10.100.3–5	3,998	10.9	0.9	–	–	–	–	–	–	4.6	3.8	17.0
10.100.4–1	4,345	28.8	1.2	–	–	–	–	–	–	0.2	0.1	50.0
10.100.4–2	4,604	26.9	1.5	–	–	–	–	–	–	0.3	0.2	33.0
10.100.4–3	4,463	26.2	1.1	–	–	–	–	–	–	0.3	0.3	0.0
10.100.4–4	5,158	25.5	1.1	–	–	–	–	–	–	0.4	0.2	50.0
10.100.4–5	4,028	25.1	1.3	–	–	–	–	–	–	0.6	0.5	16.0
10.100.5–1	4,193	35.6	1.8	–	–	–	–	–	–	0.3	0.2	33.0
10.100.5–2	4,607	36.0	1.7	–	–	–	–	–	–	0.3	0.4	−33.0
10.100.5–3	4,237	38.2	1.4	–	–	–	–	–	–	0.4	0.5	−25.0
10.100.5–4	4,799	35.3	1.3	–	–	–	–	–	–	<0.0	0.2	200.0
10.100.5–5	3,874	35.9	1.5	2.3	2.3	2.3	–	–	–	4.1	0.5	88.0

Tabla 6.14: Instancias del conjunto C con $|I^L| = |I^F| = 10$.

En la Tabla 6.14 comienza la experimentación del conjunto C de instancias, donde el número de instalaciones abiertas es de diez por líder y seguidor. Se puede ver que GRASP-TS logra encontrar la solución óptima en todas las instancias, mientras que GRASP no lo logra para la instancia 10.100.5-5. En términos de tiempos de ejecución, se observa que cuando la metaheurística encuentra el valor óptimo, se encuentra en menos de un segundo en la mayoría de los casos. Además, en 11 de las 20 instancias totales, GRASP-TS es más rápido que GRASP, pero en general, solo hay un promedio de ahorro alrededor del 28 %.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP BV	% GAP GRASP AV	% GAP GRASP WV	% GAP GRASP-TS BV	% GAP GRASP-TS AV	% GAP GRASP-TS WV	Tiempo promedio (seg.) GRASP	Tiempo promedio (seg.) GRASP-TS	Ahorro (%)
12.100.2–1	4,618	4.1	0.6	–	–	–	–	–	–	0.1	0.1	0.0
12.100.2–2	4,457	3.8	0.9	–	–	–	–	–	–	0.1	<0.0	100.0
12.100.2–3	4,271	4.3	0.8	–	–	–	–	–	–	0.2	<0.0	220.0
12.100.2–4	4,998	4.2	0.8	–	–	–	–	–	–	0.2	0.1	50.0
12.100.2–5	4,168	4.1	0.7	–	–	–	–	–	–	0.1	0.1	0.0
12.100.3–1	4,664	19.5	1.6	–	–	–	–	–	–	0.3	0.1	66.0
12.100.3–2	4,510	18.8	1.7	–	–	–	–	–	–	0.3	0.1	66.0
12.100.3–3	4,399	25.5	1.5	–	–	–	–	–	–	0.2	0.5	−150.0
12.100.3–4	5,096	21.7	1.7	–	–	–	–	–	–	0.6	0.4	33.0
12.100.3–5	4,168	20.9	1.7	–	–	–	–	–	–	0.2	0.3	−50.0
12.100.4–1	4,666	56.7	2.6	–	–	–	–	–	–	0.2	0.3	−50.0
12.100.4–2	4,576	52.3	3.4	–	–	–	–	–	–	0.7	0.6	−14.0
12.100.4–3	4,456	70.9	3.4	–	–	–	–	–	–	0.7	0.6	−14.0
12.100.4–4	5,162	64.2	2.6	–	–	–	–	–	–	2.9	0.8	72.0
12.100.4–5	4,100	54.3	2.8	–	–	–	–	–	–	0.4	0.4	0.0
12.100.5–1	4,629	113.0	4.2	–	–	–	–	–	–	0.2	0.5	−150.0
12.100.5–2	4,634	115.7	4.9	–	–	–	–	–	–	0.4	0.5	−25.0
12.100.5–3	4,286	120.8	4.4	–	–	–	–	–	–	0.9	1.2	−33.0
12.100.5–4	5,266	117.1	4.1	–	–	–	–	–	–	2.6	0.7	73.0
12.100.5–5	3,961	103.0	4.8	–	–	–	–	–	–	0.2	0.3	−50.0
12.100.6–1	4,552	152.3	5.3	–	–	–	–	–	–	0.8	0.7	13.0
12.100.6–2	4,727	155.4	5.8	–	–	–	–	–	–	0.6	0.6	0.0
12.100.6–3	4,249	155.2	5.8	–	–	–	–	–	–	10.3	1.5	85.0
12.100.6–4	5,266	149.3	5.3	–	–	–	–	–	–	1.5	0.9	40.0
12.100.6–5	3,817	140.8	5.8	–	–	–	–	–	–	7.2	3.1	57.0

Tabla 6.15: Instancias del conjunto C con $|I^L| = |I^F| = 12$.

Los resultados para las instancias con doce instalaciones por líder y seguidor del conjunto C se muestra en la Tabla 6.15. Para estas instancias observamos un comportamiento similar a las instancias con diez instalaciones pues en todas ellas el segundo método es mejor que el primero en cuanto a tiempo de ejecución al resolver a lo mucho en 5.8 segundos como vemos en las instancias dos, tres y cinco con $p = r = 6$. Se observa además que las metaheurísticas logran encontrar la solución óptima en todas las instancias en las diez corridas donde los tiempos de ejecución indican que en 15 de las 20 instancias el GRASP es más rápido que el GRASP-TS. Para la mayoría de las instancias los metaheurísticos no requieren más de un segundo para llegar a la solución óptima, salvo las instancias tipo cuatro con $p = r \in \{4, 5\}$ y las tipo tres, cuatro y cinco con $p = r \in \{6\}$.

Instancia	Valor Óptimo	García-Vélez et al. (2024) Tiempo (seg.)	Fischetti et al. (2017) Tiempo (seg.)	% GAP GRASP BV	AV	WV	% GAP GRASP-TS BV	AV	WV	Tiempo promedio (seg.) GRASP	GRASP-TS	Ahorro (%)
14.100.2–1	4,801	8.9	1.0	–	–	–	–	–	–	0.2	0.3	−50.0
14.100.2–2	4,777	6.1	1.2	–	–	–	–	–	–	<0.0	<0.0	0.0
14.100.2–3	3,990	7.7	1.1	–	–	–	–	–	–	0.4	0.4	0.0
14.100.2–4	4,989	5.8	1.1	–	–	–	–	–	–	0.2	0.2	0.0
14.100.2–5	4,311	6.1	1.1	–	–	–	–	–	–	0.2	0.3	−50.0
14.100.3–1	4,700	46.0	3.2	–	–	–	–	–	–	0.9	0.7	22.0
14.100.3–2	4,827	28.7	2.7	–	–	–	–	–	–	0.1	0.1	0.0
14.100.3–3	4,030	41.5	3.3	–	–	–	–	–	–	0.6	0.7	−16.0
14.100.3–4	4,989	37.7	3.0	–	–	–	–	–	–	0.3	0.5	−66.0
14.100.3–5	4,293	36.6	3.4	–	–	–	–	–	–	0.6	0.4	33.0
14.100.4–1	4,597	146.3	7.7	–	–	–	–	–	–	2.3	0.7	70.0
14.100.4–2	4,889	124.7	5.9	–	–	–	–	–	–	0.4	0.3	25.0
14.100.4–3	4,035	143.0	7.8	–	–	–	–	–	–	0.8	1.1	−38.0
14.100.4–4	4,866	133.2	7.2	–	–	–	–	–	–	0.6	0.8	−33.0
14.100.4–5	4,230	122.4	7.5	–	–	–	–	–	–	1.1	0.6	45.0
14.100.5–1	4,491	314.4	13.5	–	–	–	–	–	–	2.3	0.9	61.0
14.100.5–2	4,958	279.4	12.3	–	–	–	–	–	–	0.4	0.4	0.0
14.100.5–3	3,995	320.8	14.5	–	–	–	–	–	–	1.0	1.3	−30.0
14.100.5–4	4,780	289.4	14.1	–	–	–	–	–	–	3.0	1.1	63.0
14.100.5–5	4,214	290.2	16.5	–	–	–	–	–	–	2.9	1.2	58.0
14.100.6–1	4,336	506.9	20.7	–	–	–	–	–	–	2.8	0.9	68.0
14.100.6–2	5,027	500.9	17.1	–	–	–	–	–	–	2.9	2.2	24.0
14.100.6–3	3,860	522.1	25.5	–	–	–	–	–	–	9.6	5.1	47.0
14.100.6–4	4,798	489.6	22.1	–	–	–	–	–	–	1.8	1.3	28.0
14.100.6–5	4,209	480.3	21.4	–	–	–	–	–	–	4.2	3.1	26.0

Tabla 6.16: Instancias del conjunto C con $|I^L| = |I^F| = 14$.

En la Tabla 6.16 se muestran los resultados de las ultimas instancias del tipo C. Podemos observar que en estas instancias el segundo método resulta ser mucho mejor que el primero en cuanto a tiempo computacional, esto debido a que resuelve en a lo mucho 25.5 segundos en las instancias que requieren una mayor combinatoria, es decir cuando $p = r = 6$ mientras que para el primer método estas tardan a lo menos 480.3 segundos en resolver. Es evidente que ambas metaheurísticas encuentran la solución óptima en todas las instancias a lo largo de las diez ejecuciones. En términos de tiempo de ejecución, una vez más, GRASP-TS demuestra ser más rápido que GRASP en general (suma de los tiempos promedio). Sin embargo, los tiempos de ejecución no superan los 9.6 segundos para GRASP y los 5.1 segundos para GRASP-TS.

Instancia p=r	Incumbente	% GAP GRASP BV	AV	WV	% GAP GRASP-TS BV	AV	WV	Tiempo promedio (seg.) GRASP	GRASP-TS	Ahorro (%)
2	64,613	–	–	–	–	–	–	0.2	0.1	50.0
3	64,613	–	–	–	–	–	–	0.2	<0.0	250.0
4	64,613	–	–	–	–	–	–	0.4	<0.0	420.0
5	64,613	–	–	–	–	–	–	0.6	<0.0	580.0
6	64,613	–	–	–	–	–	–	1.6	0.8	50.0
7	64,613	–	–	–	–	–	–	6.3	2.7	57.0

Tabla 6.17: Caso real para diferentes valores de $p = r$.

En la Tabla 6.17 se muestran los resultados del caso real pero solamente resueltos con las metaheurísticas. Se puede observar que en ambos casos el valor incumbente coincide con la solución óptima reportada en la Tabla 6.7 del modelo abordado con lealtad, además de que en todos los casos el valor incumbente se obtiene para las diez corridas. Se observa que GRASP-TS demuestra un rendimiento superior en términos de tiempo de ejecución, requiriendo menos de 0.09 segundos para $p, r = \{3, 4, 5\}$.

Por último, se decidió crear y ejecutar instancias con 500 clientes y al menos 50 instalaciones por empresa competidora para cerrar 25, 35, 45 y 50 instalaciones. Dichos resultados se pueden observar en la Tabla 6.18. Es evidente que, debido al tamaño de las instancias, los métodos exactos no lograron obtener la solución óptima para estas instancias. Estas fueron abordadas con ambos métodos propuestos. Los resultados obtenidos por GRASP y GRASP-TS demuestran que ambos algoritmos son una opción viable para resolver este problema complejo. Como era de esperar, el tiempo requerido aumenta considerablemente, pero esto se debe al gran tamaño de las instancias. A pesar de ello, la eficiencia de las metaheurísticas parece mantenerse. En este conjunto de instancias, GRASP-TS necesitó (en promedio) menos tiempo y se observa que los tiempos registrados son aceptables para un problema estratégico de esta índole.

Instancias	Valor	% GAP GRASP			% GAP GRASP-TS			Tiempo promedio (seg)		Ahorro
grandes	Incumbente	BV	AV	WV	BV	AV	WV	GRASP	GRASP-TS	(%)
50.500.25	26,006	–	–	–	–	–	–	12.915	11.133	13.8
70.500.35	21,664	–	0.02	0.06	–	0.04	0.06	21.614	20.447	5.4
90.500.45	23,526	–	0.02	0.04	–	0.02	0.04	42.177	33.776	19.9
100.500.50	64,613	–	0.01	0.03	–	<0.00	<0.00	78.779	66.236	15.9

Tabla 6.18: Instancias de mayor tamaño.

En la Figura 6.2 se muestra el rendimiento de los algoritmos probados en términos del tiempo computacional requerido usando el modelo con preferencias, se traza el tiempo requerido para cada instancia. Como en el caso del modelo considerando lealtad, se observa de nuevo que GRASP-TS es el algoritmo que requiere menos tiempo para resolver el problema.

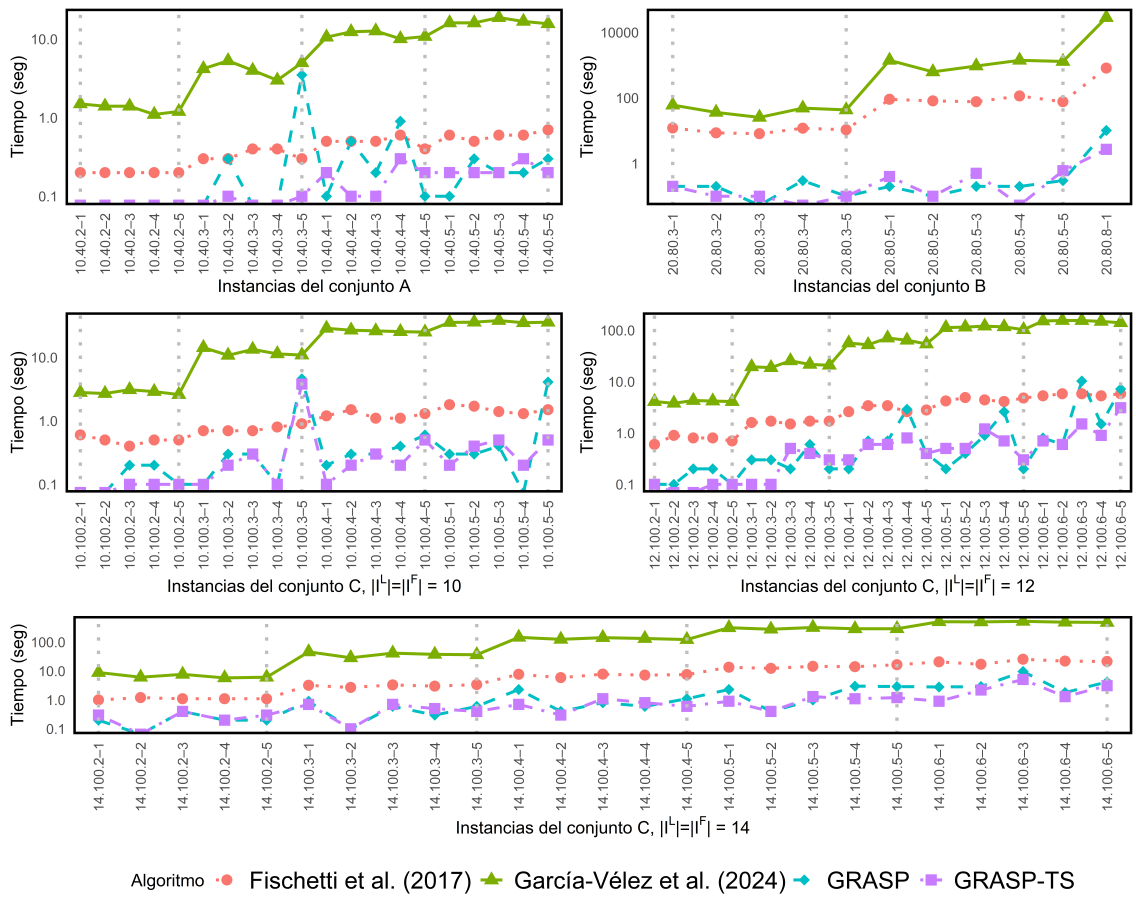


Figura 6.2: Ilustración del tiempo computacional requerido por cada algoritmo probado usando el modelo $\overline{(r|p)}^{\mathcal{P}} - Centroide$.

Capítulo 7

Sensibilidad

En esta capítulo se realiza un análisis destinado a evaluar el impacto de la lealtad a la hora de resolver el problema $\overline{(r|p)}^{\mathcal{L}} - Centroide$, es decir a la hora de cerrar instalaciones en un mercado competitivo. El cual nos indica la importancia de considerar o no la lealtad en los mercados competitivos y como se ve reflejado el porcentaje de captura de demanda de las empresas bajo diferentes niveles de lealtad por parte de los clientes. Por último describimos un acontecimiento presente en este tipo de problemas al cual se le conoce la ventaja del segundo jugador.

7.1 Evaluar el impacto de la lealtad

Las Tablas 7.1 a 7.5 mostradas en esta sección dan un resumen a los resultados de nuestros experimentos computacionales pero centrándose en la distribución final de la demanda y no en la eficiencia computacional. Todas las tablas tienen la misma estructura: en la primera columna se muestra el nombre de la instancia definidas como en la Sección 6.3, seguido de la demanda total en el mercado y la captura inicial del líder. Las siguientes dos columnas muestran la asignación de la demanda después del cierre de instalaciones, para los casos en los que la lealtad se incluyó o no en el modelo. El siguiente par de columnas presenta el cambio relativo en la captura de demanda como resultado del esfuerzo del cierre de instalaciones calculado como $\Delta\, Relativo = \left(\dfrac{CapturaFinal - CapturaInicial}{CapturaInicial}\right) \times 100\,\%$, según sea el caso, es decir, si se considera o no la lealtad. Luego, el tercer par de columnas muestran la captura del líder en términos porcentuales después del cierre de instalaciones. Finalmente, el porcentaje de clientes que se vieron afectados por la lealtad se muestra en el último par de columnas la cual representa la proporción de clientes que el líder y el seguidor habrían perdido si sus clientes no hubieran sido leales o mejor dicho si

no se hubiera considerado la lealtad en el problema.

Instancia	Demanda Total	Captura Inicial Líder	Captura Final		% Δ Relativo		% Captura Final		% Clientes Leales	
			Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Líder	Seguidor
10.40.2-1	3,533	1,510	1,510	1,510	0.0	0.0	42.7	42.7	0.0	0.0
10.40.2-2	3,812	1,863	1,938	1,863	4.0	0.0	50.8	48.9	9.5	5.3
10.40.2-3	4,232	1,078	1,078	1,078	0.0	0.0	25.5	25.5	7.1	0.0
10.40.2-4	4,207	1,629	1,629	1,629	0.0	0.0	38.7	38.7	6.7	12.0
10.40.2-5	4,001	1,969	1,879	1,969	−4.6	0.0	47.0	49.2	15.8	0.0
10.40.3-1	3,533	1,510	1,510	1,510	0.0	0.0	42.7	42.7	0.0	0.0
10.40.3-2	3,812	1,863	2,018	1,962	8.3	5.3	52.9	51.5	9.5	5.3
10.40.3-3	4,232	1,078	1,179	1,078	9.4	0.0	27.9	25.5	7.1	11.5
10.40.3-4	4,207	1,629	1,629	1,629	0.0	0.0	38.7	38.5	6.7	12.0
10.40.3-5	4,001	1,969	1,773	1,927	−9.9	−2.1	44.3	48.2	15.8	0.0
10.40.4-1	3,533	1,510	1,510	1,520	0.0	0.7	42.7	43.0	11.8	4.3
10.40.4-2	3,812	1,863	2,061	2,053	10.6	10.2	54.1	53.9	9.5	5.3
10.40.4-3	4,232	1,078	1,333	1,197	23.6	11.0	31.5	28.3	7.1	15.4
10.40.4-4	4,207	1,629	1,642	1,629	0.8	0.0	39.0	38.7	6.7	12.0
10.40.4-5	4,001	1,969	1,649	1,852	−16.3	−5.9	41.2	46.3	21.1	4.8
10.40.5-1	3,533	1,510	1,461	1,520	−3.3	0.7	41.4	43.0	11.8	4.3
10.40.5-2	3,812	1,863	2,061	2,122	10.6	13.9	54.1	55.7	14.3	5.3
10.40.5-3	4,232	1,078	1,528	1,316	41.7	22.1	36.1	31.1	14.3	11.5
10.40.5-4	4,207	1,629	1,672	1,672	2.6	2.6	39.7	39.7	6.7	12.0
10.40.5-5	4,001	1,969	1,507	1,843	−23.5	−6.4	37.7	46.1	21.1	4.8

Tabla 7.1: Instancias con 10 instalaciones por competidor y 40 clientes.

En la Tabla 7.1 se presentan los resultados de las instancias con 10 instalaciones y 40 clientes pertenecientes a los casos no estructurados, es decir del conjunto A de instancias previamente definidas en la Sección 6.1, donde las instalaciones se localizan aleatoriamente y estas no comparten la misma localización con los clientes. Los resultados indican que hay un comportamiento inestable de la lealtad en la asignación de demanda después del proceso de cierre de instalaciones, pues solamente en ocho de veinte instancias la captura inicial se mejora después del cierre de instalaciones, en nueve se mantiene y solamente en tres instancias esta captura empeora, esto último se ve reflejado en los Δ relativos negativos. Los hallazgos no muestran una relación clara entre el número de instalaciones cerradas y la dirección de los cambios en la forma en que se distribuye la demanda entre el líder y el seguidor. En este conjunto de casos no estructurados, también es difícil encontrar evidencia de que la lealtad favorezca consistentemente al líder o seguidor, pues el porcentaje de clientes leales al líder y seguidor no rebasa el 21.1 y 12 %, respectivamente. Todas estas deducciones pueden deberse al hecho de que tanto los clientes como las instalaciones están bastante dispersos en el territorio, lo que hace poco probable que el cliente encuentre una instalación de su proveedor preferido dentro de su radio de lealtad. Por otro lado, con un mayor número de instalaciones y clientes más concentrados geográficamente, aumenta la probabilidad de tener las instalaciones de un proveedor favorito dentro del radio de lealtad del cliente.

Instancia	Demanda Total	Captura Inicial Líder	Captura Final		% Δ Relativo		% Captura Final		% Clientes Leales	
			Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Líder	Seguidor
10.100.2-1	10,359	4,550	4,985	4,485	9.6	−1.4	48.1	43.3	0.0	19.6
10.100.2-2	10,148	4,542	4,883	4,595	7.5	1.2	48.1	45.3	9.1	10.7
10.100.2-3	9,411	4,181	4,666	4,477	11.6	7.1	49.6	47.6	4.9	11.9
10.100.2-4	10,252	5,029	5,588	5,298	11.1	5.4	54.5	51.7	6.7	12.7
10.100.2-5	8,917	3,974	4,270	3,974	7.5	0.0	47.9	44.6	2.1	19.2
10.100.3-1	10,359	4,550	4,891	4,481	7.5	−1.5	47.2	43.3	0.0	23.2
10.100.3-2	10,148	4,542	4,784	4,604	5.3	1.4	47.1	45.4	6.8	17.9
10.100.3-3	9,411	4,181	4,676	4,500	11.8	7.6	49.7	47.8	4.9	16.9
10.100.3-4	10,252	5,029	5,526	5,389	9.9	7.2	53.9	52.6	15.6	14.5
10.100.3-5	8,917	3,974	4,266	3,998	7.4	0.6	47.8	44.8	6.3	13.5
10.100.4-1	10,359	4,550	4,887	4,345	7.4	−4.5	47.2	41.9	4.5	26.8
10.100.4-2	10,148	4,542	4,697	4,604	3.4	1.4	46.3	45.4	11.4	23.2
10.100.4-3	9,411	4,181	4,557	4,463	9.0	6.7	48.4	47.4	7.3	18.6
10.100.4-4	10,252	5,029	5,304	5,158	5.5	2.6	51.7	50.3	15.6	16.4
10.100.4-5	8,917	3,974	4,108	4,028	3.4	1.4	46.1	45.2	8.3	21.2
10.100.5-1	10,359	4,550	4,670	4,193	2.6	−7.9	45.1	40.5	6.8	28.6
10.100.5-2	10,148	4,542	4,528	4,607	−0.3	1.4	44.6	45.4	11.4	23.2
10.100.5-3	9,411	4,181	4,388	4,237	5.0	1.3	46.6	45.0	7.3	22.0
10.100.5-4	10,252	5,029	4,997	4,799	−0.6	−4.6	48.7	46.8	11.1	25.5
10.100.5-5	8,917	3,974	4,105	3,874	3.3	−2.5	46.0	43.5	8.3	25.0

Tabla 7.2: Instancias con 10 instalaciones por competidor y 100 clientes.

En la Tabla 7.2 se muestran los resultados a las instancias con 100 clientes y 10 instalaciones pertenecientes al conjunto de instancias B las cuales caen en el caso estructurado. Se observa que al considerar lealtad del total de 20 instancias, en una la captura del líder se mantiene, en 13 se mejora y solamente en 6 esta empeora, vemos que la captura del líder adopta un mejor comportamiento si no se considera la lealtad, por lo que bajo este comportamiento parece ser que si tomamos en cuenta la lealtad de los clientes esta tiende a favorecer al seguidor, lo cual podemos corroborar en el último par de columnas pues es evidente como el mayor número de clientes son leales al seguidor pues se tiene porcentajes de clientes leales que oscilan entre un 10.5 % y 28.6 % en todos los casos, mientras que el líder este alcanza a lo más 15.6 %

Instancia	Demanda Total	Captura Inicial Líder	Captura Final		% Δ Relativo		% Captura Final		% Clientes Leales	
			Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Líder	Seguidor
12.100.2-1	10,359	4,618	5,013	4,618	8.6	0.0	48.4	44.6	0.0	13.2
12.100.2-2	10,148	4,457	4,841	4,457	8.6	0.0	47.7	43.9	4.5	10.7
12.100.2-3	9,411	4,147	4,586	4,271	10.6	3.0	48.7	45.4	5.1	13.1
12.100.2-4	10,252	4,878	5,475	4,998	12.2	2.5	53.4	48.8	10.4	9.6
12.100.2-5	8,917	4,168	4,431	4,168	6.3	0.0	49.7	46.7	2.1	13.5
12.100.3-1	10,359	4,618	5,043	4,664	9.2	1.0	48.7	45.0	2.1	24.5
12.100.3-2	10,148	4,457	4,857	4,510	9.0	1.2	47.9	44.4	4.5	16.1
12.100.3-3	9,411	4,147	4,752	4,399	14.6	6.1	50.5	46.7	10.3	13.1
12.100.3-4	10,252	4,878	5,694	5,096	16.7	4.5	55.5	49.7	14.6	19.2
12.100.3-5	8,917	4,168	4,422	4,168	6.1	0.0	49.6	46.7	4.2	19.2
12.100.4-1	10,359	4,618	5,241	4,666	13.5	1.0	50.6	45.0	2.1	26.4
12.100.4-2	10,148	4,457	4,890	4,576	9.7	2.7	48.2	45.1	6.8	21.4
12.100.4-3	9,411	4,147	4,752	4,456	14.6	7.5	50.5	47.4	10.3	21.3
12.100.4-4	10,252	4,878	5,620	5,162	15.2	5.8	54.8	50.4	18.8	21.2
12.100.4-5	8,917	4,168	4,353	4,100	4.4	−1.6	48.8	46.0	10.4	23.1
12.100.5-1	10,359	4,618	5,269	4,629	14.1	0.2	50.9	44.7	4.3	28.3
12.100.5-2	10,148	4,457	4,788	4,634	7.4	4.0	47.2	45.7	9.1	21.4
12.100.5-3	9,411	4,147	4,739	4,286	14.3	3.4	50.4	45.5	10.3	29.5
12.100.5-4	10,252	4,878	5,607	5,266	14.9	8.0	54.7	51.4	20.8	23.1
12.100.5-5	8,917	4,168	4,305	3,961	3.3	−5.0	48.3	44.4	12.5	23.1
12.100.6-1	10,359	4,618	5,259	4,552	13.9	−1.4	50.8	43.9	10.6	34.0
12.100.6-2	10,148	4,457	4,742	4,727	6.4	6.1	46.7	46.6	11.4	21.4
12.100.6-3	9,411	4,147	4,894	4,249	18.0	2.5	52.0	45.2	15.4	27.9
12.100.6-4	10,252	4,878	5,325	5,266	9.2	8.0	51.9	51.4	22.9	15.4
12.100.6-5	8,917	4,168	4,166	3,817	−0.1	−8.4	46.7	42.8	12.5	32.7

Tabla 7.3: Instancias con 12 instalaciones por competidor y 100 clientes.

Para la Tabla 7.3 vemos que el líder se ve favorecido si resuelve sin lealtad pues en todos los casos la captura inicial mejoro desde un 3.3 % hasta un 18 %, solamente en una instancia el líder presento perdida de demanda con respecto a su captura inicial. Por otro lado, tomando en cuenta la lealtad se ve una tendencia a la baja en cuanto a la captura final del líder, pues en la mayoría de los casos presenta un leve cambio a comparación de resolver sin lealtad, incluso en cuatro instancias el líder presenta perdida, por lo que de nueva cuenta podemos llegar a la consolación de que el seguidor se ve beneficiado si se considera la lealtad de los clientes. En el último par de columnas vemos reflejado como el seguidor se favorece cuando existe lealtad a comparación del líder.

Instancia	Demanda Total	Captura Inicial Líder	Captura Final		% Δ Relativo		% Captura Final		% Clientes Leales	
			Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Líder	Seguidor
14.100.2-1	10,359	4,822	5,274	4,801	9.4	−0.4	50.9	46.4	6.8	10.7
14.100.2-2	10,148	4,777	5,166	4,777	8.1	0.0	50.9	47.1	2.2	9.1
14.100.2-3	9,411	3,980	4,222	3,990	6.1	0.3	44.9	42.4	5.7	7.7
14.100.2-4	10,252	4,989	5,447	4,989	9.2	0.0	53.1	48.7	8.5	5.7
14.100.2-5	8,917	4,311	4,447	4,311	3.2	0.0	49.9	48.4	14.6	13.5
14.100.3-1	10,359	4,822	5,304	4,700	10.0	−2.5	51.2	45.4	6.8	12.5
14.100.3-2	10,148	4,777	5,181	4,827	8.5	1.1	51.1	47.6	6.7	12.7
14.100.3-3	9,411	3,980	4,334	4,030	8.9	1.3	46.1	42.8	5.7	15.4
14.100.3-4	10,252	4,989	5,447	4,989	9.2	0.0	53.1	48.7	12.8	9.4
14.100.3-5	8,917	4,311	4,378	4,293	1.6	−0.4	49.1	48.1	10.4	21.2
14.100.4-1	10,359	4,822	5,294	4,597	9.8	−4.7	51.1	44.4	6.8	14.3
14.100.4-2	10,148	4,777	5,191	4,889	8.7	2.3	51.2	48.2	13.3	16.4
14.100.4-3	9,411	3,980	4,410	4,035	10.8	1.4	46.9	42.9	5.7	16.9
14.100.4-4	10,252	4,989	5,486	4,866	10.0	−2.5	53.5	47.5	12.8	13.2
14.100.4-5	8,917	4,311	4,307	4,230	−0.1	−1.9	48.3	47.4	16.7	23.1
14.100.5-1	10,359	4,822	5,256	4,491	9.0	−6.9	50.7	43.4	6.8	23.2
14.100.5-2	10,148	4,777	5,269	4,958	10.3	3.8	51.9	48.9	13.3	21.8
14.100.5-3	9,411	3,980	4,494	3,995	12.9	0.4	47.8	42.5	8.6	21.5
14.100.5-4	10,252	4,989	5,519	4,780	10.6	−4.2	53.8	46.6	14.9	20.8
14.100.5-5	8,917	4,311	4,214	4,214	−2.3	−2.3	47.3	47.3	20.8	25.0
14.100.6-1	10,359	4,822	5,355	4,336	11.1	−10.1	51.7	41.9	6.8	28.6
14.100.6-2	10,148	4,777	5,408	5,027	13.2	5.2	53.3	49.5	22.2	21.8
14.100.6-3	9,411	3,980	4,421	3,860	11.1	−3.0	47.0	41.0	11.4	21.5
14.100.6-4	10,252	4,989	5,481	4,798	9.9	−3.8	53.5	46.8	21.3	28.3
14.100.6-5	8,917	4,311	4,113	4,209	−4.6	−2.4	46.1	47.2	31.3	26.9

Tabla 7.4: Instancias con 14 instalaciones por competidor y 100 clientes.

Por último, en Tabla 7.4 se observa el mismo comportamiento que para las instancias con 10 y 12 instalaciones, el líder se ve favorecido si no se considera la lealtad, pues solo en tres instancias del total presenta una pérdida de demanda, mientras que con lealtad esto se cumple en 13 instancias, es decir, de nuevo se nota una tendencia a la baja en cuanto a la captura final del líder, pues esta sigue manteniéndose o bien presentando un leve cambio en la captura final. Esto se observa en el penúltimo par de columnas, donde vemos que considerando la lealtad la captura final es menor que no considerándola, es decir parece ser que la demanda del seguidor se ve beneficiada si se considera la lealtad de los clientes.

De manera general, de las Tablas 7.2 a 7.4 donde se muestran los resultados para las instancias estructuradas, es decir las del conjunto B, podemos concluir que cuando el número de establecimientos cerrados es el mismo $p = r$, la lealtad tiende a favorecer al seguidor y una posible explicación es que la lealtad puede inducir algún tipo de *ventaja del segundo jugador*, una característica común de los modelos de localización líder-seguidor[1]

[1]En modelos de localización competitivos, el seguidor siempre puede capturar al menos el 50 % del mercado localizando sus instalaciones junto con las del líder; esto se ha denominado en la literatura la *ventaja del segundo jugador*. Véase, por ejemplo, Serra and ReVelle (1994b), quienes discuten esta situación.

Instancia	Demanda Total	Captura Inicial Líder	Captura Final		% Δ Relativo		% Captura Final		% Clientes Leales	
			Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Sin Lealtad	Lealtad	Líder	Seguidor
9.150.2	138,432	63,266	64,831	62,231	2.5	−1.6	46.8	45.0	4.5	12.0
9.150.4	138,432	63,266	69,237	62,455	9.4	−1.3	50.0	45.1	11.9	18.1
12.150.2	138,432	60,277	63,026	61,098	4.6	1.4	45.5	44.1	4.8	17.0
12.150.4	138,432	60,277	65,520	60,888	8.7	1.0	47.3	44.0	12.9	20.5
12.150.6	138,432	60,277	63,070	59,486	4.6	−1.3	45.6	43.0	48.4	10.2
15.150.2	138,432	62,157	63,154	62,566	1.6	0.7	45.6	45.2	9.5	5.7
15.150.4	138,432	62,157	63,226	63,867	1.7	2.8	45.7	46.1	17.5	11.5
15.150.6	138,432	62,157	62,996	62,398	1.4	0.4	45.5	45.1	25.4	17.2
20.150.2	138,432	62,424	64,667	61,500	3.6	−1.5	46.7	44.4	4.6	7.1
20.150.4	138,432	62,424	65,045	60,581	4.2	−3.0	47.0	43.8	9.2	15.3
20.150.6	138,432	62,424	64,040	59,379	2.6	−4.9	46.3	42.9	16.9	15.3

Tabla 7.5: Instancias con 150 clientes basada en la red de Swain.

Por último, se presenta en la tabla 7.5 los resultados de las instancias provenientes de la Red de Swain que tiene 150 clientes y se cuentan con 9, 12, 15 o 20 instalaciones. Se observa estas instancias siguen adoptando un comportamiento que beneficia al seguidor si se resuelve considerando lealtad, pues en la mayoría de los casos el $\Delta\, Relativo$ es negativo o este es muy pequeño a comparación de resolver sin lealtad, donde se aprecia que en todos los casos la captura final se mejoró.

Para finalizar, se pueden observar dos características importantes que confirman lo observado en los ejemplos anteriores: a) cuando los esfuerzos de cierre de instalaciones se realizan simultáneamente y no hay lealtad, el líder parece tener ventaja y su captura de demanda aumenta en la mayoría de los casos; y b) con la lealtad del cliente, la situación se invierte y el seguidor parece ser capaz de capturar porciones mayores de la demanda en el mercado. Los resultados de esta sección brindan evidencia que respalda la conjetura de que la lealtad espacial puede desempeñar un papel en la distribución del mercado entre empresas que participan simultáneamente en procesos de cierre de instalaciones. Por lo tanto, es importante evaluar cuál será el resultado de tomar decisiones de cierre de instalaciones sin considerar la lealtad del cliente. Por lo que esto se aborda en la siguiente sección.

7.2 ¿Importa la lealtad?

En la sección anterior vimos como en la mayoría de las instancias el líder pierde participación de mercado (demanda) si se toma en cuenta la lealtad de los clientes para resolver el problema, se esperaría que este enfrentara mayores pérdidas de demanda si ignorara la lealtad de los clientes al decidir qué instalaciones cerrar, pero pasa lo contrario, cuando ignora la lealtad se presenta un aumento considerable en su participación de mercado. Es por eso que con el fin de responder entonces la siguiente pregunta: ¿Importa la lealtad? en esta sección, evaluamos el efecto de tomar (o no tomar la lealtad) en consideración de las reacciones de los clientes ante

las decisiones del líder y seguidor.

Para probar esto, diseñamos el siguiente experimento, cuyos resultados se resumen en la Tabla 7.6. Consideramos el caso en el que tanto el líder como el seguidor tienen 14 instalaciones abiertas en una red con 100 nodos de demanda o clientes. Se resolvieron los siguientes problemas para valores de p y r del 2 al 6:

1. El problema $\overline{(r|p)}-Centroide$ se resuelve para ambos competidores asumiendo que el radio de lealtad es cero. Esto se representa en la Tabla 7.6 donde usamos la notación $\widehat{\rho}^L = \widehat{\rho}^F = 0$. Una vez determinadas las instalaciones a cerrar, se asigna la demanda a las instalaciones que permanecen abiertas utilizando un radio de lealtad $\rho = 2$ para ambos competidores.

2. El problema $\overline{(r|p)} - Centroide$ se resuelve asumiendo que el líder ignora la lealtad, $\widehat{\rho}^L = 0$; y que el seguidor reconoce la existencia de lealtad, es decir, $\widehat{\rho}^F = 2$. La asignación de la demanda se calcula considerando $\rho = 2$.

3. El problema $\overline{(r|p)} - Centroide$ se resuelve asumiendo que el líder considera la lealtad, $\widehat{\rho}^L = 2$; y que el seguidor la ignora, es decir $\widehat{\rho}^F = 0$. La asignación de la demanda se calcula considerando $\rho = 2$.

4. El problema $\overline{(r|p)} - Centroide$ se resuelve asumiendo que ambos competidores tienen en cuenta la lealtad en su proceso de decisión, es decir, $\widehat{\rho}^L = \widehat{\rho}^F = 2$. Por lo tanto la asignación se toma considerando lealtad en ambos competidores.

p=r	Configuración		Captura de Demanda (%)			
			14.100.p-1		14.100.p-2	
			Líder	Seguidor	Líder	Seguidor
-	Captura Inicial		46.55 %	53.45 %	47.07 %	52.93 %
2	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 0$	46.52 %	53.48 %	50.75 %	49.25 %
	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 2$	46.35 %	53.65 %	47.07 %	52.93 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 0$	46.52 %	53.48 %	50.75 %	49.25 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 2$	46.35 %	53.65 %	47.07 %	52.93 %
3	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 0$	46.93 %	53.07 %	49.52 %	50.48 %
	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 2$	45.37 %	54.63 %	47.57 %	52.43 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 0$	46.93 %	53.07 %	51.27 %	48.73 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 2$	45.37 %	54.63 %	47.57 %	52.43 %
4	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 0$	46.83 %	53.17 %	51.25 %	48.75 %
	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 2$	44.38 %	55.62 %	47.68 %	52.32 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 0$	46.83 %	53.17 %	52.65 %	47.35 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 2$	44.38 %	55.62 %	48.18 %	51.82 %
5	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 0$	45.95 %	54.05 %	51.99 %	48.01 %
	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 2$	43.35 %	56.65 %	48.86 %	51.14 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 0$	45.95 %	54.05 %	51.25 %	48.75 %
	$\widehat{\rho}^L = 2$	$p^F = 2$	43.35 %	56.65 %	48.86 %	51.14 %
6	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 0$	42.63 %	57.37 %	53.36 %	46.64 %
	$\widehat{\rho}^L = 0$	$\widehat{\rho}^F = 2$	41.86 %	58.14 %	49.29 %	50.71 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 0$	42.63 %	57.37 %	51.99 %	48.01 %
	$\widehat{\rho}^L = 2$	$\widehat{\rho}^F = 2$	41.86 %	58.14 %	49.54 %	50.46 %

Tabla 7.6: Distribución del mercado cuando se ignora la lealtad.

En la Tabla 7.6 se muestra en la primera columna el número de instalaciones a cerrar donde $p = r$, seguido de la configuración con la cual se resolvió la instancia, es decir si se tomó no en cuenta la lealtad del líder o seguidor, por último, se muestra el porcentaje de captura de demanda capturada por el líder o seguidor dependiendo el tipo de instancia. Los resultados sugieren que el principal beneficio de tener en cuenta la lealtad del cliente va al seguidor pues en los cinco casos donde $\widehat{\rho}^L = \widehat{\rho}^F = 2$ este siempre mejoro su captura de demanda a comparación de la asignación inicial, más aún, apoyando la hipótesis de la existencia de la ventaja del segundo tomador de decisiones: en todos los casos, la participación de mercado capturada por el seguidor es mayor o se mantiene igual a la captura inicial cuando la lealtad del cliente se incluye en su problema de cierre de instalaciones. Asimismo, aunque esto no es significativo y sólo se puede observar en un para estas dos instancias, parece que el líder también puede beneficiarse de tener en cuenta la lealtad del cliente a la hora de emprender procesos de cierre.

7.3 El impacto de la asignación inicial

En esta sección presentamos un conjunto adicional de experimentos donde la asignación inicial de clientes se desvía de la práctica estándar de la instalación más cercana. El objetivo es ilustrar que tanto el modelo y los métodos de solución arrojan resultados razonables independientemente de la asignación inicial de los clientes. Comparamos cuatro escenarios, que consisten en una asignación determinista y tres aleatorias: en el primero, los clientes prefieren su establecimiento más cercano; en las aleatorias, las probabilidades de elegir una instalación líder son del 30 %, 50 % y 70 %, respectivamente. En todos estos casos, el radio de lealtad se fija en un valor constante, que es igual a los radios promedio utilizados en cada subconjunto de experimentos en las Tablas 6.16 y 7.4.

También incluimos un conjunto adicional de experimentos en los cuales los clientes se asignan inicialmente a la instalación más cercana y el radio de lealtad se proporciona como un factor $h_0 = 1.24\,\%$ de la distancia desde cada nodo hasta la instalación más cercana del proveedor no preferido (d_{jc}). Esta configuración nos permite modelar una situación en la que el cliente está dispuesto a viajar una fracción adicional de la distancia a la instalación disponible más cercana para llegar a su proveedor preferido, en lugar de una distancia absoluta.

La Tabla 7.7 muestra los resultados obtenidos para varios experimentos donde la asignación inicial de preferencias varía. Las primeras dos columnas muestran los radios fijos de lealtad utilizados y la demanda total asociada a cada instancia. Los cuatro pares de columnas siguientes muestran el porcentaje de demanda captura-

da por el líder en función de la estrategia de asignación (instalación más cercana, 30 %- %70, 50 %-50 % y 70 %-30 %, respectivamente), y la captura final de mercado resultante después de cerrar el número de instalaciones indicadas por la etiqueta de las instancias. La última columna de la tabla muestra los resultados obtenidos cuando el radio de lealtad se expresa como un porcentaje h_0 de la distancia al competidor más cercano. La tabla 7.9 muestra resultados similares pero para el caso asimétrico, es decir donde cada empresa cierra un número diferente de instalaciones.

Como podemos observar en ambos casos, no es posible identificar un patrón en los resultados que pueda sugerir la existencia de algún tipo de ventaja, ni para el líder ni para el seguidor. Nuestros resultados sugieren que la distribución final de la captura de demanda depende en gran medida de la configuración original de la red y de las interacciones espaciales entre las empresas y sus clientes.

Instancia	Radio de Lealtad	Demanda Total	Asignación Usual (%)		Asignación Aleatoria (%)						$\rho = h_0\, d_{jc}$
			Más Cercano	Captura Final	30-70	Captura Final	50-50	Captura Final	70-30	Captura Final	Captura Final
14.100.2-1				46.4		41.2		52.7		58.7	49.2
14.100.3-1				45.4		42.5		53.1		56.5	51.0
14.100.4-1	0.346	10,359	46.5	44.4	41.2	43.8	52.9	52.0	58.9	57.1	52.9
14.100.5-1				44.0		43.5		51.0		56.4	53.2
14.100.6-1				43.6		44.5		49.7		55.8	53.4
14.100.2-2				47.1		44.3		54.3		61.6	51.0
14.100.3-2				47.8		44.8		54.8		61.0	51.7
14.100.4-2	0.345	10,148	47.1	47.8	44.3	45.9	54.3	55.4	62.1	60.3	52.4
14.100.5-2				48.3		45.1		55.4		59.6	52.6
14.100.6-2				48.6		45.6		55.8		59.1	52.8
14.100.2-3				41.8		43.4		54.3		54.7	44.7
14.100.3-3				40.6		43.4		53.4		53.3	46.9
14.100.4-3	0.340	9,411	42.3	39.1	43.4	42.4	54.9	52.4	55.9	51.8	48.7
14.100.5-3				37.4		40.9		51.2		50.0	49.1
14.100.6-3				37.3		41.6		49.8		49.6	48.7
14.100.2-4				48.7		41.9		54.7		59.9	50.2
14.100.3-4				48.7		41.2		54.2		59.9	52.0
14.100.4-4	0.349	10,252	48.7	47.8	41.9	41.5	54.7	53.9	59.9	58.8	54.1
14.100.5-4				47.5		43.1		53.6		57.5	54.2
14.100.6-4				46.5		42.8		52.3		56.5	54.2
14.100.2-5				48.2		42.5		53.8		56.8	49.7
14.100.3-5				47.0		41.9		52.2		55.6	50.9
14.100.4-5	0.354	8,917	48.4	46.2	42.6	40.5	55.9	50.8	58.9	53.7	52.3
14.100.5-5				45.4		40.2		49.1		51.8	53.5
14.100.6-5				44.5		40.3		47.4		50.3	52.6

Tabla 7.7: Distribución de mercado con asignaciones iniciales alternativas, y $p = r$.

Instancia	Radio de Lealtad	Demanda Total	Asignación Usual (%)		Asignación Aleatoria (%)						$\rho = h_0 d_{jc}$
			Más Cercano	Captura Final	30-70	Captura Final	50-50	Captura Final	70-30	Captura Final	Captura Final
14.100.3.5-1				46.83		45.52		55.65		59.75	55.46
14.100.5.3-1	0.3458	10,359	46.55	43.35	41.16	40.62	52.9	0.51	58.93	55.01	48.74
14.100.4.6-1				47.25		47.08		54.83		59.95	58.22
14.100.6.4-1				41.65		39.99		46.95		52.95	48.08
14.100.3.5-2				52.58		47.03		56.89		62.76	57.47
14.100.5.3-2	0.3447	10,148	47.07	44.23	44.32	42.7	54.26	53.46	62.12	58.83	47.28
14.100.4.6-2				53.34		49.6		58.36		62.68	58.21
14.100.6.4-2				42.65		41.56		53.33		57.02	46.9
14.100.3.5-3				40.68		43.37		53.41		53.34	52.35
14.100.5.3-3	0.3398	9,411	42.29	37.31	43.37	40.94	54.89	51.21	55.88	49.97	43.58
14.100.4.6-3				41.37		44.62		52.97		54.01	55.12
14.100.6.4-3				35.41		39.42		49.69		48.05	42.38
14.100.3.5-4				49.92		43.74		56.33		60.46	57.37
14.100.5.3-4	0.3489	10,252	48.66	46.16	41.91	39.36	54.74	50.74	59.89	55.96	49.3
14.100.4.6-4				51.3		46.82		57.49		61.67	60.31
14.100.6.4-4				43.39		37.73		48.66		52.66	48.27
14.100.3.5-5				50.25		43.49		55.93		58.17	56.12
14.100.5.3-5	0.3544	8,917	48.35	42.14	42.64	38.87	55.92	46.07	58.92	49.31	48.23
14.100.4.6-5				50.96		44.75		54.79		56.89	58.04
14.100.6.4-5				40.5		36.85		43.41		46.85	46.42

Tabla 7.8: Distribución del mercado con asignaciones iniciales alternativas, y $p \neq r$.

Instancia	Radio de Lealtad	Demanda Total	Asignación Usual (%)		Asignación Aleatoria (%)						$\rho = h_0 d_{jc}$
			Más Cercano	Captura Final	30-70	Captura Final	50-50	Captura Final	70-30	Captura Final	Captura Final
14.100.3.5-1				46.83		45.52		55.65		59.75	55.46
14.100.5.3-1	0.3458	10,359	46.55	43.35	41.16	40.62	52.9	0.51	58.93	55.01	48.74
14.100.4.6-1				47.25		47.08		54.83		59.95	58.22
14.100.6.4-1				41.65		39.99		46.95		52.95	48.08
14.100.3.5-2				52.58		47.03		56.89		62.76	57.47
14.100.5.3-2	0.3447	10,148	47.07	44.23	44.32	42.7	54.26	53.46	62.12	58.83	47.28
14.100.4.6-2				53.34		49.6		58.36		62.68	58.21
14.100.6.4-2				42.65		41.56		53.33		57.02	46.9
14.100.3.5-3				40.68		43.37		53.41		53.34	52.35
14.100.5.3-3	0.3398	9,411	42.29	37.31	43.37	40.94	54.89	51.21	55.88	49.97	43.58
14.100.4.6-3				41.37		44.62		52.97		54.01	55.12
14.100.6.4-3				35.41		39.42		49.69		48.05	42.38
14.100.3.5-4				49.92		43.74		56.33		60.46	57.37
14.100.5.3-4	0.3489	10,252	48.66	46.16	41.91	39.36	54.74	50.74	59.89	55.96	49.3
14.100.4.6-4				51.3		46.82		57.49		61.67	60.31
14.100.6.4-4				43.39		37.73		48.66		52.66	48.27
14.100.3.5-5				50.25		43.49		55.93		58.17	56.12
14.100.5.3-5	0.3544	8,917	48.35	42.14	42.64	38.87	55.92	46.07	58.92	49.31	48.23
14.100.4.6-5				50.96		44.75		54.79		56.89	58.04
14.100.6.4-5				40.5		36.85		43.41		46.85	46.42

Tabla 7.9: Distribución del mercado con asignaciones iniciales alternativas, y $p \neq r$.

Capítulo 8

Conclusiones y Trabajo a futuro

En este capítulo se presentan las conclusiones al definir y resolver un problema bi-nivel de cierre de instalaciones competitivas considerando lealtad de los clientes y se menciona de manera general los resultados obtenidos. También se muestran las líneas de investigación futuras que se deben desarrollar para continuar la investigación mostrada en este trabajo de tesis.

8.1 Conclusiones

En el Capítulo 1 se presentó la introducción de esta tesis doctoral la cual considera un problema donde dos empresas que brindan un servicio a un conjunto de clientes buscan cerrar algunas instalaciones que tienen operando con normalidad debido a que ya no es rentable mantenerlas en operación. Se menciona que entre estas empresas existe una jerarquía de mando, o bien una de ellas tiene mejor posicionamiento en el mercado que otra y por ende se aborda como un problema de cierre de instalaciones competitivas el cual fue modelado haciendo uso de la programación bi-nivel. Como novedad, se consideró el uso de lealtad por parte de los clientes en lugar de asignarse a la instalación más cercana.

Una extensa revisión de literatura sobre problemas de cierre de instalaciones y además de localización de instalaciones competitivas en los que se han utilizado algoritmos exactos, heurísticos y metaheurísticos se mostró en el Capítulo 2. Notemos que existe poca literatura sobre problemas que abordan el cierre de instalaciones pues hasta donde se sabe existen solo los trabajos de Wang et al. (2003), ReVelle et al. (2007), Bhaumik (2010), Ruiz-Hernández et al. (2015), Ruiz-Hernández and Delgado-Gómez (2016), Ruiz-Hernández et al. (2017) y producto de esta tesis nace el trabajo de García-Vélez et al. (2024).

En el Capítulo 3 se mostró el modelo matemático de cierre de instalaciones competitivas considerando lealtad de los clientes a quien denotamos como el problema $\overline{(r|p)}^{\mathcal{L}}$- *Centroide* y a su vez se presentó un modelo matemático equivalente que considera las preferencias de los clientes denotado como el problema $\overline{(r|p)}^{\mathcal{P}} - Centroide$. Con el nuevo modelo se logra eliminar las variables de asignación z y las restricciones que la involucran, teniendo así un modelo con menos restricciones y variables.

En el Capítulo 4 presentamos una breve revisión de literatura sobre los métodos exactos disponibles en programación bi-nivel, donde evidenciamos que existen la forma exhaustiva, los métodos de descomposición, los de ramificación y acotamiento y los de ramificación y corte todos ellos con la ventaja de encontrar la solución óptima pero con la principal desventaja de que para instancias que requieren un alta combinatoria estos requieren más esfuerzo computacional. También se describen dos métodos exactos que utilizamos para resolver los problemas presentados en el Capítulo 3, uno llamado "Algoritmo de enumeración implícita de búsqueda en profundidad" el cual es una adaptación a nuestro problema de un algoritmo propuesto por Bard and Moore (1992) con la principal diferencia en la regla de corte y otro llamado simplemente como "Algoritmo genérico para resolver problemas bi-nivel entero-mixto" propuesto en Fischetti et al. (2017) el cual utiliza un problema de un solo nivel llamado *HPR* que al resolver de manera iterativa ayuda a generar cortes.

Como bien mencionamos anteriormente una desventaja al resolver nuestro problema con un método exacto es que solo podremos hacerlo para instancias de tamaño limitado, por lo que con el fin de poder resolver instancias que requieren una mayor combinatoria a bajo costo computacional se desarrollaron e implementaron dos metaheurísticas: un GRASP y una híbrida que con el fin de intensificar la búsqueda de las soluciones intercambia la fase de búsqueda local por una búsqueda tabú denominada GRASP-TS, ambos métodos se pueden ver en el Capítulo 5.

En el Capítulo 6 mostramos los resultados numéricos obtenidos para tres conjuntos de instancias y un caso real para los cuales se conoce la solución óptima pues resolvimos utilizando los métodos exactos mencionados en el Capítulo 4. Adicional a esto, creamos 59 instancias de prueba que requieren una mayor combinatoria para las cuales no se conoce el valor optimo. Al resolver las instancias utilizando el modelo con lealtad, se observa que en la mayoría de ellas el algoritmo propuesto por Fischetti et al. (2017) es computacionalmente más rápido en tiempo. Por otro lado, los resultados con las metaheurísticas son prometedores pues además de encontrar el valor óptimo en todos los casos, estas logran reducir considerablemente el tiempo de ejecución. Al usar el modelo donde se consideran las preferencias de los clientes, los resultados indican que al eliminar considerablemente muchas variables y restricciones este resuelve en menor tiempo para ambos casos, es decir tanto métodos exactos

como metaheurísticos.

Por último, en el Capítulo 7 se llevó a cabo una serie de experimentos que nos indican la importancia de considerar la lealtad a la hora de resolver el problema que presentamos en esta tesis sobre el cierre de instalaciones. Además, se describió una peculiaridad de este tipo de problemas que entran en competencia al surgir la denominada ventaja del segundo jugador, pues en todas las instancias el más beneficioso a la hora de capturar la demanda final resultaba siempre más favorable para el segundo tomador de decisiones es decir el seguidor.

8.2 Trabajo a futuro

Como hemos visto en la Sección 6 los resultados numéricos usando metaheurísticas parecen ser buenas alternativas para resolver los problemas bajo estudio en esta tesis, por lo que una linea directa de investigación seria probar algunos otros algoritmos de esta índole como por ejemplo métodos genéticos, evolutivos, búsqueda dispersa, enjambre de partículas, entre otros, para una vista más general leer Camacho-Vallejo et al. (2023).

Otra área que se puede explorar es la versión estocástica de los problemas $\overline{(r|p)}^{\mathcal{L}}$- *Centroide* y $\overline{(r|p)}^{\mathcal{P}}-$*Centroide*, principalmente en la función objetivo, es decir considerar la incertidumbre de la demanda que puede llegar a tener un cliente al cual se le brindas un servicio, pues se sabe que día a día el mercado es muy volátil y es natural pensar que el cliente puede cambiar la cantidad de demanda a adquirir de un momento a otro. Además, se sabe que para resolver un problema mediante programación estocástica este resuelve de manera óptima escenarios del problema para luego promediar los resultados obtenidos, para ello se necesita un método para resolver esos escenarios de forma óptima, por lo que en este trabajo de tesis ya propusimos dos formas de resolver a optimalidad nuestros modelos.

Los problemas de cierre de instalaciones se han puesto de moda en los últimos años y como hemos mencionado en esta sección es evidente que existe futuro sobre este tipo de problemas que se vuelven día a día más retadores e interesantes.

Apéndice A

Tutorial acerca del algoritmo de Fischetti et al. (2017)

Como bien mencionamos en el Capítulo 4, el algoritmo que se propone en Fischetti et al. (2017) no es tan sencillo de implementar, sin embargo, es un método exacto que resuelve problemas bi-nivel entero-mixto. Con el fin de dar un panorama general al lector en este apéndice se muestra a manera de tutorial como se hace esta ejecución.

Para poder llevar a cabo la implementación, a continuación describo lo que necesitamos:

1. Un ordenador con el sistema operativo Linux (Por ejemplo Ubuntu).
2. Las librerías de CPLEX cargadas en el sistema operativo mencionado.
3. Los paquetes del algoritmo de Fischetti et al. (2017) los cuales nos ayudaran a generar las librerías dinámicas de CPLEX (1-libilocplex.so, 2-libcplex.so y 3-libconcert.so).
4. Una licencia para poder usar el algoritmo de Fischetti et al. (2017) la cual se obtiene solicitándola a los autores del mismo vía correo.
5. Convertir las instancias de ejecución en un archivo al estilo "Mathematical Programming System" conocido simplemente como formato (MPS) con su respectivo archivo auxiliar cuyo formato es (AUX).
6. Añadir ambos archivos en la carpeta de ambiente y ejecutar.

Se deja al lector investigar cómo llevar a cabo los puntos 1 y 2 ya que se puede encontrar mucha información relevante en las páginas oficiales de dichos requerimientos. Para los puntos 3 y 4 en este enlace puede encontrar información sobre

dichos pasos `https://msinnl.github.io/pages/bilevel.html`. Sin embargo, no hay mucha información sobre como ejecutar el paso 5, por lo que a continuación vamos a mostrar a manera de tutorial como convertir una instancia pequeña en formato .mps y .aux.

Para iniciar el algoritmo primero debemos estar familiarizados con los formatos MPS y AUX. El formato MPS es utilizado para representar problemas de programación matemática en forma estándar. Es un formato de archivo que define un conjunto de variables, restricciones y objetos matemáticos (algo así como el formato LP (.lp)) el cual consta de tres secciones principales: la sección "NAME" (nombre), la sección "ROWS" (filas) y la sección "COLUMNS" (columnas). La sección "NAME" proporciona un nombre descriptivo para el problema. La sección "ROWS" se encarga de enumerar las filas de la matriz de restricciones, especificando si son restricciones de igualdad, de menor igual o mayor igual. La sección "COLUMNS" enumera las columnas de la matriz, donde se definen las variables y sus coeficientes en las restricciones y la función objetivo. Además de estas secciones principales, un archivo MPS también contiene una sección llamada RHS"(lados derechos), que especifica los valores en el lado derecho de las restricciones, y "BOUNDS"(cotas), donde se define los rangos permitidos para las variables y también donde se indican las cotas de las variables. Mientras que el formato AUX es un archivo que contiene datos adicionales que son necesarios para que un optimizador pueda interpretar y leer de manera correcta el formato MPS, ya que contiene información sobre cuantas variables y restricciones se tienen en el modelo y además se indica cuáles son los índices de las mismas. En nuestro contexto el formato AUX indica el número de variables y restricciones del nivel inferior, así como el índice de las variables y restricciones del mismo. Para exportar un modelo en formato MPS los optimizadores comerciales incluyen instrucciones para poder exportar un modelo en dicho formato, sin embargo es el archivo en formato AUX el cual se debe construir, en base a la tabla A.1:

Indicador	**Descripción**
N	Especifica el número de variables de nivel inferior en el modelo.
M	Especifica el número de restricciones de nivel inferior en el modelo.
LC	Especifica el índice de las variables del nivel inferior. Nota: se supone que las variables están en el orden en que aparecen en el archivo MPS; índices comienzan en 0.
LR	Especifica el índice de las restricciones del nivel inferior. Nota: se supone que las restricciones están en el orden en que aparecen en el archivo MPS; índices comienzan en 0.
LO	Especifica los coeficientes de la función objetivo del nivel inferior. Nota: se supone que los coeficientes están en el orden de las variables de nivel inferior.
OS	Especifica el sentido objetivo de nivel inferior (1=minimización, -1=maximización).

Tabla A.1: Indicadores del formato AUX.

A continuación mostraremos un ejemplo práctico de un archivo.mps y un archivo.aux, considerando el siguiente modelo de programación bi-nivel presentado en Bard (2013) donde $(x^*, y^*) = (2, 2)$ con $F(x^*, y^*) = -22$:

$$\min_{x} \quad -x-10y \tag{R4}$$

$$\text{Sujeto a:} \quad \min_{y} \quad y \tag{1}$$

$$\text{Sujeto a:} \quad -25x+20y \leq 30 \tag{R0}$$

$$x+2y \leq 10 \tag{R1}$$

$$2x-y \leq 15 \tag{R2}$$

$$2x+10y \geq 15 \tag{R3}$$

$$x, y \geq 0, \quad x, y \in \mathbb{Z}_+ \tag{2}$$

Para el modelo bi-nivel entero-mixto anterior tenemos que el ***archivo.mps*** es el siguiente:

Bard.mps:

1. NAME Modelo_Entero_Mixto
2. ROWS
3. L $R0$
4. L $R1$
5. L $R2$
6. G $R3$
7. N $R4$
8. COLUMNS
9. x $R0$ -25
10. x $R1$ 1
11. x $R2$ 2
12. x $R3$ 2
13. x $R4$ -1
14. y $R0$ 20
15. y $R1$ 2
16. y $R2$ -1
17. y $R3$ 10
18. y $R4$ -10
19. RHS
20. B $R0$ 30
21. B $R1$ 10
22. B $R2$ 15
23. B $R3$ 15
24. BOUNDS
25. LI BOUND x
26. LI BOUND y
27. ENDATA

Para tener un panorama más claro sobre este tipo de archivos, favor de visitar el siguiente link `https://www-eio.upc.edu/lceio/manuals/cplex-11/pdf/reffileformatscplex.pdf`.

Por último, siguiendo la nomenclatura de la tabla A.1 tenemos que el ***archivo.aux*** es:

Bard.aux:

1. N 1
2. M 4
3. LC 1
4. LR 0
5. LR 1
6. LR 2
7. LR 3
8. LO 1
9. OS 1

Teniendo estos dos archivos (***Bard.mps*** y ***Bard.aux***) en la misma carpeta en que tenemos el ejecutable del algoritmo de Fischetti et al. (2017) junto con la licencia, ejecutamos la siguiente instrucción:

```
charly@charly-garcia:~/bilevelBinary$ ./bilevel -mpsfile bard.mps -setting 99
```

Figura A.1: Instrucción en terminal para ejecutar Bard.mps.

y al dar enter resuelve obteniendo la siguiente información:

```
>>>> Final solution: leader objective value -22.0000000000 (cplex value -22.0000000000 truecost -22.0000000000) time       0.04 <<<<


 OK: feasibility check passed with cost -22.000000 and f* 2.0000000000000000000 vs Phi(x*) 2.0000000000000000000
 OK, solved to optimality :-))

STAH; input_file ; zbest ; final_bound ; root_bound; time (s.); root_time (s.); opt ;nodes ; %root_gap; %final_gap; setting
STAT; bard.mps ; -22.000000 ; -22.000000  ; -22.000000 ; 0.041471 ; 0.040003 ; 1 ;0 ; 0.000000 ; 0.000000 ; 99


-----------------------------------------------------------------------------------------------------------------------
 BEST SOLUTION AVAILABLE FOR bard.mps
 LEADER COST -22.000000 FOLLOWER COST 2.000000

                    x                 2.0000000000
                    y                 2.0000000000
-----------------------------------------------------------------------------------------------------------------------
```

Figura A.2: Solución a Bard.mps.

En la cual nos reporta la solución óptima $(x^*, y^*) = (2, 2)$ con $F(x^*, y^*) = -22$.

Resumen autobiográfico

Juan-Carlos García-Vélez

Candidato para obtener el grado de
Doctorado en Ciencias con Orientación en Matemáticas

Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico-Matemático

Tesis:
Un problema bi-nivel de cierre de instalaciones competitivas: Alternativas de modelación y solución.

Nací el 03 de julio de 1993 en Monterrey, Nuevo León, México. Soy el primer hijo del sr *Juan Carlos García Hernández* y la sra *Leticia Vélez Rojas*. Obtuve mi título de Licenciatura en Matemáticas en la Facultad de Ciencias Físico Matemáticas (FCFM) de la Universidad Autónoma de Nuevo León (UANL) en diciembre de 2016. Terminé mis estudios de maestría en enero de 2019 en el Programa de Posgrado en Ciencias con Orientación en Matemáticas de la FCFM-UANL con el apoyo del entonces Consejo Nacional de Ciencias y Tecnología (CONACYT). Posteriormente, en el mismo mes de enero de 2019, inicié mis estudios de doctorado basado en problemas de cierre de instalaciones competitivos modelados con programación bi-nivel. Durante estos tres escalones de estudios he estado bajo el mando y dirección del Dr. *José-Fernando Camacho-Vallejo*. Mi área de investigación es la optimización combinatoria específicamente en la definición de problemas de localización y/o cierre de instalaciones competitivas modelados con programación bi-nivel y a su vez diseñar técnicas de solución eficientes para resolver dichos problemas.

Bibliografía

Abdinnour-Helm, S. and Hadley, S. W. (2000). Tabu search based heuristics for multi-floor facility layout. *International Journal of Production Research*, 38(2):365–383.

Ahmed, S., Ahmed, S., and Ahmed, T. (2018). A branch-and-cut-and-price algorithm for bi-level integer programming. *European Journal of Operational Research*, 270(3):1051–1063.

Alekseeva, E., Kochetov, Y., and Plyasunov, A. (2015). An exact method for the discrete $(r|p)$-centroid problem. *Journal of Global Optimization*, 63(3):445–460.

Alekseeva, E., Kochetova, N., Kochetov, Y., and Plyasunov, A. (2010). Heuristic and exact methods for the discrete $(r|p)$-centroid problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 11–22. Springer.

Bard, J. F. (2013). *Practical bilevel optimization: algorithms and applications*, volume 30. Springer Science & Business Media.

Bard, J. F. and Moore, J. T. (1992). An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL)*, 39(3):419–435.

Benati, S. and Laporte, G. (1994). Tabu search algorithms for the (r/xp)-medianoid and (r/p)-centroid problems. *CENTRE DE RECHERCHE SUR LES TRANSPORTS PUBLICATION*, (941).

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

Bhadury, J., Eiselt, H. A., and Jaramillo, J. H. (2003). An alternating heuristic for medianoid and centroid problems in the plane. *Computers & Operations Research*, 30(4):553–565.

Bhaumik, P. K. (2010). Optimal shrinking of the distribution chain: the facilities delocation decision. *International Journal of Systems Science*, 41(3):271–280.

Biesinger, B., Hu, B., and Raidl, G. (2014). An evolutionary algorithm for the leader-follower facility location problem with proportional customer behavior. In *International Conference on Learning and Intelligent Optimization*, pages 203–217. Springer.

Biesinger, B., Hu, B., and Raidl, G. (2015). A hybrid genetic algorithm with solution archive for the discrete $(r|p)$-centroid problem. *Journal of Heuristics*, 21(3):391–431.

Blum, C. (2010). **Hybrid Metaheuristics**. *Computers & Operations Research*, 37(3):430 – 431. Hybrid Metaheuristics.

Calvete, H. I., Galé, C., Iranzo, J. A., Camacho-Vallejo, J.-F., and Casas-Ramírez, M.-S. (2020). A matheuristic for solving the bilevel approach of the facility location problem with cardinality constraints and preferences. *Computers & Operations Research*, 124:105066.

Camacho-Vallejo, J.-F., Casas-Ramírez, M., and Miranda, P. (2014a). The p-median bilevel problem under preferences of the customers. *Recent Advances in Theory, Methods and Practice of Operations Research*, pages 121–127.

Camacho-Vallejo, J.-F., Cordero-Franco, Á. E., and González-Ramírez, R. G. (2014b). Solving the bilevel facility location problem under preferences by a stackelberg-evolutionary algorithm. *Mathematical Problems in Engineering*, 2014.

Camacho-Vallejo, J.-F., Corpus, C., and Villegas, J. G. (2023). Metaheuristics for bilevel optimization: A comprehensive review. *Computers & Operations Research*, page 106410.

Campos-Rodríguez, C., Moreno-Pérez, J. A., and Santos-Peñate, D. R. (2012). Particle swarm optimization with two swarms for the discrete $(r|p)$-centroid problem. In *International Conference on Computer Aided Systems Theory*, pages 432–439. Springer.

Candler, W. and Norton, R. (1977). *Multi-level programming and development policy*. The World Bank.

Carrizosa, E., Davydov, I., and Kochetov, Y. (2012). A new alternating heuristic for the $(r|p)$–centroid problem on the plane. In *Operations Research Proceedings 2011*, pages 275–280. Springer.

Casas-Ramírez, M.-S. and Camacho-Vallejo, J.-F. (2017). Solving the p-median bilevel problem with order through a hybrid heuristic. *Applied Soft Computing*, 60:73–86.

Casas-Ramírez, M.-S., Camacho-Vallejo, J.-F., Díaz, J. A., and Luna, D. E. (2017). A bi-level maximal covering location problem. *Operational Research*, pages 1–29.

Casas-Ramírez, M.-S., Camacho-Vallejo, J.-F., and Martínez-Salazar, I.-A. (2018). Approximating solutions to a bilevel capacitated facility location problem with customer's patronization toward a list of preferences. *Applied Mathematics and Computation*, 319:369–386.

Chen, J., Gao, L., and Wang, J. (2018). A survey of exact algorithms for solving the bi-level programming problem. *Journal of Industrial and Management Optimization*, 14(2):401–420.

Davydov, I., Kochetov, Y., and Carrizosa, E. (2014a). A local search heuristic for the $(r|p)$-centroid problem in the plane. *Computers & operations research*, 52:334–340.

Davydov, I. A., Kochetov, Y. A., Mladenovic, N., and Urosevic, D. (2014b). Fast metaheuristics for the discrete $(r|p)$-centroid problem. *Automation and Remote Control*, 75(4):677–687.

Delmaire, H., Díaz, J. A., Fernández, E., and Ortega, M. (1999). Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *INFOR: Information Systems and Operational Research*, 37(3):194–225.

Dempe, S. and Zemkoho, A. B. (2011). An exact algorithm for solving bi-level linear programs. *Journal of Global Optimization*, 51(2):221–245.

Díaz, J. A., Luna, D. E., Camacho-Vallejo, J.-F., and Casas-Ramírez, M.-S. (2017). Grasp and hybrid grasp-tabu heuristics to solve a maximal covering location problem with customer preference ordering. *Expert Systems with Applications*, 82:67–76.

Díaz, J. A., Luna, D. E., and Zetina, C. A. (2013). A hybrid algorithm for the manufacturing cell formation problem. *Journal of Heuristics*, 19(1):77–96.

Drezner, Z. and Hamacher, H. W. (2001). *Facility location: applications and theory*. Springer Science & Business Media.

Duarte, A. and Martí, R. (2007). Tabu search and grasp for the maximum diversity problem. *European Journal of Operational Research*, 178(1):71–84.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

Fischetti, M., Ljubić, I., Monaci, M., and Sinnl, M. (2017). A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, 65(6):1615–1637.

Fischetti, M., Ljubić, I., Monaci, M., and Sinnl, M. (2018). On the use of intersection cuts for bilevel optimization. *Mathematical Programming*, 172(1):77–103.

García-Vélez, J.-C. (2019). *El problema (r|p)-Centroide negativo considerando lealtad de los clientes*. PhD thesis, Universidad Autónoma de Nuevo León.

García-Vélez, J.-C., Ruiz-Hernández, D., Camacho-Vallejo, J.-F., and Díaz, J. A. (2024). Competitive network restructuring with spatially loyal customers. a bilevel facility delocation problem. *Computers & Operations Research*, page 106418.

Garfinkel, R. and Zhao, Q. (2008). Branch-and-bound algorithms for solving bilevel linear programming problems. *INFORMS Journal on Computing*, 20(3):385–397.

Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206.

Glover, F. (1990). Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32.

Hakimi, S. L. (1983). On locating new facilities in a competitive environment. *European Journal of Operational Research*, 12(1):29–35.

Hotelling, H. (1929). Stability in competition. *The Economic Journal*, 39(153):41–57.

Kochetov, Y., Kochetova, N., and Plyasunov, A. (2013). A matheuristic for the leader-follower facility location and design problem. In *Proceedings of the 10th Metaheuristics International Conference (MIC 2013)*, volume 32. Citeseer.

Kress, D. and Pesch, E. (2012a). $(r|p)$-centroid problems on networks with vertex and edge demand. *Computers & Operations Research*, 39(12):2954–2967.

Kress, D. and Pesch, E. (2012b). Sequential competitive location on networks. *European Journal of Operational Research*, 217(3):483–499.

Laguna, M. and Velarde, J. L. G. (1991). A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent manufacturing*, 2(4):253–260.

Laporte, G., Nickel, S., and Saldanha-da Gama, F. (2019). Introduction to location science. In *Location science*, pages 1–21. Springer.

Lim, A. and Wang, F. (2004). A smoothed dynamic tabu search embedded grasp for m-vrptw. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 704–708. IEEE.

Lodi, A., Ralphs, T. K., and Woeginger, G. J. (2014). Bilevel programming and the separation problem. *Mathematical Programming*, 146(1-2):437–458.

Lozano, L. and Smith, J. C. (2017a). A backward sampling framework for interdiction problems with fortification. *INFORMS Journal on Computing*, 29(1):123–139.

Lozano, L. and Smith, J. C. (2017b). A value-function-based exact approach for the bilevel mixed-integer programming problem. *Operations Research*, 65(3):768–786.

Lozano, M. and García-Martínez, C. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers & Operations Research*, 37(3):481–497.

Marić, M., Stanimirović, Z., and Milenković, N. (2012). Metaheuristic methods for solving the bilevel uncapacitated facility location problem with clients' preferences. *Electronic Notes in Discrete Mathematics*, 39:43–50.

MirHassani, S., Raeisi, S., and Rahmani, A. (2015). Quantum binary particle swarm optimization-based algorithm for solving a class of bi-level competitive facility location problems. *Optimization Methods and Software*, 30(4):756–768.

Nezakati, H., Kuan, Y. L., and Asgari, O. (2011). Factors influencing customer loyalty towards fast food restaurants. In *International Conference on Sociality and Economics Development*, volume 10, pages 12–16.

Rehman, A., Zia ur Rehman, D., Akhtar, W., et al. (2012). Factors affecting brand loyalty: a perspective of fast food restaurants. *Rehman, A., Zia-ur-Rehman, M., & Akhtar, W.(2012). Factors affecting Brand Loyalty: a perspective of fast food restaurants. Actual Problems of Economics*, 130:13–20.

ReVelle, C., Murray, A. T., and Serra, D. (2007). Location models for ceding market share and shrinking services. *Omega*, 35(5):533–540.

Roboredo, M. C. and Pessoa, A. A. (2013). A branch-and-cut algorithm for the discrete $(r|p)$-centroid problem. *European Journal of Operational Research*, 224(1):101–109.

Rodríguez, C. M. C., Pérez, J. A. M., Noltemeier, H., and Peñate, D. R. S. (2009). Two-swarm pso for competitive location problems. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, pages 115–126. Springer.

Ruiz-Hernández, D. and Delgado-Gómez, D. (2016). The stochastic capacitated branch restructuring problem. *Annals of Operations Research*, 246(1-2):77–100.

Ruiz-Hernández, D., Delgado-Gómez, D., and López-Pascual, J. (2015). Restructuring bank networks after mergers and acquisitions: A capacitated delocation model for closing and resizing branches. *Computers & Operations Research*, 62:316–324.

Ruiz-Hernández, D., Elizalde, J., and Delgado-Gómez, D. (2017). Cournot–stackelberg games in competitive delocation. *Annals of Operations Research*, 256(1):149–170.

Serra, D. and Colomé, R. (2001). Consumer choice and optimal locations models: formulations and heuristics. *Papers in Regional Science*, 80(4):439–464.

Serra, D. and ReVelle, C. (1994a). Competitive location in discrete space.

Serra, D. and ReVelle, C. (1994b). Market capture by two competitors: the preemptive location problem. *Journal of regional Science*, 34(4):549–561.

Shahsavari, M. and Kuhn, D. (2017). A mixed-integer linear programming formulation and benders decomposition for the bi-level programming problem with sub-gradient lower level problems. *Journal of Global Optimization*, 69(2):355–377.

Stackelberg, H. v. et al. (1952). Theory of the market economy.

Swoboda, B., Berg, B., Schramm-Klein, H., and Foscht, T. (2013). The importance of retail brand equity and store accessibility for store loyalty in local competition. *Journal of Retailing and Consumer Services*, 20(3):251–262.

Tam, L., Wood, W., and Ji, M. F. (2009). Brand loyalty is not habitual. *Handbook of brand relationships*, pages 43–62.

Tamura, K. and Nakayama, H. (2018). A bi-level optimization approach based on benders decomposition for the bus transit network design problem. *Transportation Research Part C: Emerging Technologies*, 89:135–152.

Tang, Y., Richard, J.-P. P., and Smith, J. C. (2016). A class of algorithms for mixed-integer bilevel min–max optimization. *Journal of Global Optimization*, 66:225–262.

Venter de Villiers, M., Visnenza, A., and Phiri, N. (2018). Importance of location and product assortment on flea market loyalty. *The Service Industries Journal*, 38(11-12):650–668.

Wang, L. and Xu, P. (2017). The watermelon algorithm for the bilevel integer linear programming problem. *SIAM Journal on Optimization*, 27(3):1403–1430.

Wang, Q., Batta, R., Bhadury, J., and Rump, C. M. (2003). Budget constrained location problem with opening and closing of facilities. *Computers & Operations Research*, 30(13):2047–2069.

Weber, A. (1929). *Theory of the Location of Industries*. University of Chicago Press.

Wolf, G. W. (2011). Facility location: concepts, models, algorithms and case studies. series: Contributions to management science: edited by zanjirani farahani, reza and hekmatfar, masoud, heidelberg, germany, physica-verlag, 2009, 549 pp.,€ 171.15, $219.00,£ 144.00, isbn 978-3-7908-2150-5 (hardprint), 978-3-7908-2151-2 (electronic).

Yuan, Y., Huang, J., and Liu, S. (2016). A benders decomposition approach for solving bi-level linear programming problems with general upper level constraints. *Optimization Letters*, 10(4):849–862.