UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS

CENTRO DE INVESTIGACIÓN EN CIENCIAS FÍSICO-MATEMÁTICAS

EL PROBLEMA DE LA P-MEDIANA CON ORDEN: REFORMULACIONES Y ALGORITMOS

POR:

MARTHA SELENE CASAS RAMÍREZ

EN OPCIÓN AL GRADO DE:

MAESTRÍA EN CIENCIAS CON ORIENTACIÓN EN MATEMÁTICAS

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN MAYO 2014

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS

CENTRO DE INVESTIGACIÓN EN CIENCIAS FÍSICO-MATEMÁTICAS

EL PROBLEMA DE LA P-MEDIANA CON ORDEN: REFORMULACIONES Y ALGORITMOS

POR:

MARTHA SELENE CASAS RAMÍREZ

EN OPCIÓN AL GRADO DE:

MAESTRÍA EN CIENCIAS CON ORIENTACIÓN EN MATEMÁTICAS

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN MAYO 2014

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS

CENTRO DE INVESTIGACIÓN EN CIENCIAS FÍSICO-MATEMÁTICAS

Los miembros del Comité de Tesis recomendamos que la Tesis "El Problema de la p-Mediana con Orden: Reformulaciones y Algoritmos", realizada por la Lic. Martha Selene Casas Ramírez con número de matrícula 1327876, sea aceptada para su defensa como opción al grado de Maestría en Ciencias con Orientación en Matemáticas.

El Comité de Tesis

_____

Dr. José Fernando Camacho Vallejo

Asesor

_____ _____

Dr. Pablo Andrés Miranda González — Co-asesor

Dra. Nataliya Kalashnykova — Revisor

Vo. Bo.

_____

Dr. José Fernando Camacho Vallejo

Coordinador del Posgrado en Ciencias con Orientación en Matemáticas

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN MAYO 2014

“Si tienes un sueño en tu corazón,

y de verdad crees en él,

corres el riesgo de que

se convierta en realidad”

Walt Disney

ÍNDICE GENERAL

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.

Primeramente quiero agradecer a Dios por haberme acompañado y guiado a lo largo de mi vida, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida llena de aprendizajes, experiencias y sobre todo felicidad. Por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante toda la maestría. Infinitamente gracias por todas las bendiciones que ha derramado sobre mí y porque hizo realidad este sueño anhelado.

Le doy muchas gracias a mi familia, que son una bendición de Dios, por apoyarme en todo momento incondicionalmente. A mi mamá Margarita Ramírez Muñoz que desde pequeña me enseñó que la vida no es fácil y porque con sus sacrificios me dio la posibilidad de tener una excelente educación con la cual he podido llegar a hasta donde estoy. Porque sacrificó gran parte de su vida para formarme, por todo el tiempo que le robé pensando en mí y porque nunca podré pagar todos sus desvelos, ni aún con las riquezas más grandes del mundo. A mi hermana Claudia Marlene Casas Ramírez por su apoyo, por sus consejos, por compartir mis tristezas y alegrías y por siempre darme ánimo. Gracias a las dos por motivarme y darme la mano cuando sentía que el camino se terminaba, porque de

forma incondicional entendieron mis ausencias y mis malos momentos y porque nunca han dejado de animarme a continuar con mi formación. Por enseñarme a no rendirme y por impulsarme a realizar mis sueños. Gracias a mi Fer que a pesar de que llegó a mi vida de manera inesperada me ha traído muchas alegrías, por motivarme a ser mejor persona y por sacar la mejor versión de mí. Muchas gracias por ser parte de mi vida, los amo.

Quedo especial y sinceramente agradecida con mi asesor, el Dr. Fernando Camacho, por haber confiado en mi persona y aceptar realizar esta tesis bajo su dirección. Por brindarme la oportunidad de recurrir a su capacidad y experiencias de investigación y por su generosidad de compartir sus conocimientos conmigo. Pero no solamente quiero agradecer por este período de tesis, sino por lo que antecede y trasciende a este período. Fue mi profesor en la licenciatura y desde ahí sentí un respeto y admiración profunda a su labor. Gracias por invitarme en el servicio social porque con eso confirmé mi gusto por la optimización e investigación de operaciones. Por 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 tener tendida siempre una mano amiga. Es imposible describir en unas líneas el grado de agradecimiento que siento, así como de admiración por su labor docente, investigadora y calidad humana.

Le agradezco al Dr. Pablo Miranda por la colaboración brindada en esta tesis y haberme recibido con los brazos abiertos en mi estancia de investigación en la Pontificia Universidad Católica de Valparaíso. Por el tiempo que me pudo dedicar, por sus sugerencias e ideas y por los nuevos conocimientos que me transmitió. Gracias a mis revisores: Dr. Pablo Miranda y Dra. Nataliya Kalashnykova 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.

También les doy las gracias a las personas que hicieron mi estancia en Chile de lo más agradable: Dra. Rosa González, Dr. Luis Ascencio y Dr. Gabriel Gutiérrez. Gracias por su hospitalidad y amabilidad, hicieron mi estadía muy amena.

Gracias a Rafael Muñoz por haber sido compañero de maestría y amigo, por haber tenido la paciencia necesaria y por motivarme a seguir en momentos de frustración. Por los momentos buenos y malos que nos tocó compartir y por haber hecho un excelente equipo de trabajo conmigo.

Quiero darles las gracias a mis amigos que siempre estuvieron motivándome a seguir y que nunca dejaron que me derrumbara: Sarahí Báez, Óscar Ovalle, Cristina Villanueva y Diego Hernández. En especial, gracias a Adriana Arias, Jorge Garza y al Dr. Eduardo Cordero por siempre darme ánimo, por estar conmigo en los buenos y malos momentos, por apoyarme en mis decisiones y compartir conmigo sus conocimientos y su amistad incondicional.

Gracias al Centro de Investigación de Ciencias Físico-Matemáticas (CICFIM) por aceptarme en su programa de posgrado, a la Facultad de Ciencias Físico-Matemáticas (FCFM) y a mi Alma Mater la Universidad Autónoma de Nuevo León (UANL). Gracias al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca brindada para realizar mis estudios de posgrado y también gracias por la beca mixta hacia el extranjero que me otorgaron con la cual llevé a cabo una estancia de investigación en la Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile.

En general, quisiera hacer extensivo mi más sincero agradecimiento a todas y cada una de las personas que han vivido conmigo la realización de esta tesis y que de una u otra forma colaboraron o participaron. Les agradezco el haberme brindado todo el apoyo, colaboración, ánimo y sobre todo cariño y amistad.

RESUMEN

En el presente trabajo de tesis se analiza un modelo de programación binivel para el problema de la *p*-Mediana donde consideramos las preferencias de los clientes. Tomar en cuenta dichas preferencias es de suma importancia debido a la competencia actual del mercado. El problema que analizamos es una extensión del problema simple de localización de plantas con orden, dicha extensión se basa en que asumimos que un número predeterminado de plantas debe ser adecuadamente localizado. Suponemos además que una planta puede abastecer la demanda de varios clientes pero un cliente debe ser abastecido por una planta y que los clientes establecen una lista ordenada de preferencias indicando sus deseos de ser servidos por las plantas abiertas.

El problema se formula como un modelo de programación binivel donde el objetivo del líder es minimizar los costos de instalación y de distribución; y el seguidor quiere optimizar las preferencias ordenadas de los clientes. El objetivo del problema es abrir un número conocido de plantas que minimicen el costo total (instalación y distribución) de localización y, a su vez, minimicen las preferencias ordenadas de los clientes.

Nosotros proponemos dos reformulaciones del problema estudiado debido a la dificultad derivada de resolver los problemas de programación binivel. La primer reformulación se basa en las relaciones primal-dual del problema del nivel inferior y la segunda en la adición de un conjunto de restricciones que asegura que se sigue considerando las preferencias de los clientes. Se llevó a cabo experimentación

numérica y se mostró que las reformulaciones no son capaces de resolver las instancias de tamaño mediano en un tiempo computacional razonable. Este hecho nos llevó a desarrollar un algoritmo basado en la metaheurística Scatter Search donde se considera el equilibrio de Stackelberg durante el proceso. Durante el procedimiento iterativo de construcción de soluciones del problema binivel, el hecho de considerar la solución óptima del seguidor para cada solución del líder refleja la intención de encontrar el equilibrio de Stackelberg. Dicho algoritmo heurístico obtiene soluciones de buena calidad para todas las instancias analizadas en tiempos menores que el requerido para la solución de las reformulaciones de un solo nivel.

CAPÍTULO 1

INTRODUCCIÓN

En este capítulo se introduce de manera general el problema clásico de localización de plantas, también se describe la motivación de esta tesis y el objetivo central de la misma.

1.1 DESCRIPCIÓN DEL PROBLEMA

El problema de localización de plantas tiene su origen con el problema de Weber en el cual se quiere determinar el punto que minimice la suma de las distancias euclidianas entre ese punto y un número de puntos dados, una explicación más detallada se encuentra en Weber [1]. El problema clásico de localización de plantas es uno de los problemas más importantes de la teoría de la localización y ha atraído la atención de investigadores convirtiéndose en el tema central de un gran número de artículos de investigación. Algunas revisiones de literatura muy interesantes sobre este tema se pueden ver en Drezner y Hamacher [2], Sahin y Süral [3] y Melo et al. [4].

En el problema de localización de plantas existe un conjunto de clientes que se distribuyen en un espacio predefinido y desean que sus demandas particulares de algún determinado servicio sean satisfechas por una o varias plantas. El problema consiste en determinar dónde deben ubicarse las plantas y cómo deben asignarse los clientes con el objetivo de minimizar los costos de instalación y de distribución asociados con esa decisión en particular. Esta situación se conoce como el problema

clásico de localización de plantas (FLP por sus siglas en inglés). El problema simple de localización de plantas (SPLP por sus siglas en inglés) surge cuando las plantas tienen una capacidad infinita y pueden satisfacer todas las demandas de cualquier cliente, el modelo para este problema fue propuesto por Kuehn y Hamburger [5]. Una taxonomía de los problemas de localización y modelos se proporciona en Daskin [6], donde clasifican a los modelos en cuatro grupos: analíticos, continuos, discretos y de redes. Además, presentan ejemplos de cada grupo.

La descripción del SPLP se presenta a continuación. Sean i las plantas y j los clientes, donde $i \in I$ y $j \in J$. Los $c_{i\,j}$ representan los costos de que la planta i abastezca toda la demanda del cliente j. El costo fijo de instalación de la planta i es f_i. Las variables de decisión del problema están representadas por: las x_{ij} que representan el hecho de que la planta i satisfaga la demanda del cliente j y las y_i que representan si se instala la planta i o no. Está fácil ver que ambas variables de decisión son binarias.

El modelo matemático del problema de localización de plantas sin capacidades es:

$$\min \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{1.1}$$

sujeto a:

$$\sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{1.2}$$

$$x_{ij} \le y_i \qquad \forall i \in I, j \in J \tag{1.3}$$

$$x_{ij} \in \{0,1\}, y_i \in \{0,1\} \qquad \forall i \in I, j \in J \tag{1.4}$$

donde (1.1) es la función objetivo que desea minimizar la suma de los costos de instalación y de distribución, en (1.2) se asegura que cada cliente sea abastecido solamente por una planta, en (1.3) se garantiza la asignación de los clientes únicamente a las plantas abiertas y finalmente, (1.4) es la restricción de signo de las variables de decisión del problema.

El problema considerado en esta tesis es un problema de localización de plantas donde se tiene un número fijo de plantas por abrir, los clientes tienen preferencias hacia ciertas plantas que se deben tomar en cuenta y las plantas tienen capacidad infinita. Cuando el número total de plantas por abrir es conocido, entonces el problema es llamado de la p-Mediana (ver Hakimi [7]). Denotaremos en adelante al

problema que analizamos en esta tesis como el p-Mediana BPO (esto es, el Problema Binivel de la p-Mediana con Orden).

1.2 MOTIVACIÓN Y JUSTIFICACIÓN

El problema de localización de plantas ha sido ampliamente estudiado y por tal motivo existe un gran número de artículos que analizan y resuelven dicho problema con diferentes metodologías, dentro de las herramientas más utilizadas están las reformulaciones al modelo matemático y el diseño e implementación de métodos heurísticos. En ambas estrategias se busca encontrar un equilibrio entre el tiempo de cómputo y la calidad de las soluciones.

La programación binivel ha servido para modelar problemas de logística tomando en cuenta algunos componentes involucrados en el entorno del problema en específico que antes no se consideraban de manera conjunta. Es decir, en lugar de considerar de forma independiente dos procesos relacionados entre sí, ahora se consideran al mismo tiempo pero respetando una jerarquía en el orden de la toma de decisiones.

Por tal motivo, la programación binivel es una adecuada forma de modelarlo ya que existe una jerarquía en el problema donde el nivel superior es el costo total de localización y las preferencias de los clientes son el nivel inferior. Dicha jerarquización es debido a la importancia de considerar las preferencias de los clientes pero no es más significativa que el factor de costo total, es decir, el costo de instalación de la planta más el costo de distribución de las plantas a los clientes.

En problemas de localización, es importante considerar la opinión de los clientes al abrir nuevas plantas debido a que en muchos casos reales los clientes son libres de satisfacer su demanda en alguna planta. Usualmente esta decisión está basada en costos, preferencias, algún contrato predeterminado, un coeficiente de lealtad, entre otros.

Nosotros proponemos un par de reformulaciones con el fin de justificar la modelación binivel del problema que estudiamos aquí, la primera reformulación está basada en las relaciones primal-dual del problema del seguidor, y en la segunda reformulación se agrega un conjunto de restricciones para asegurar que se sigue considerando las preferencias de los clientes. También proponemos un algoritmo heurístico para resolver el problema formulado como un modelo binivel donde justificamos su uso debido al tiempo computacional que requieren ambas reformulaciones.

Además la escasez de trabajos del p-Mediana BPO también nos motivó a realizar esta investigación ya que después de una extensa búsqueda sólo encontramos un artículo (Alekseeva y Kochetov [16]) en donde consideran el problema de la p-Mediana con preferencias ordenadas como un modelo binivel. Es por ello que el desarrollo de esta tesis presenta un gran aporte dando a conocer otra forma de abordar el problema.

1.3 OBJETIVO

El objetivo de esta tesis consiste en estudiar el p-Mediana BPO y proponer una metodología de solución que proporcione soluciones de buena calidad a un bajo costo computacional.

Proponemos dos reformulaciones y un algoritmo heurístico debido a la dificultad existente de resolver el p-Mediana BPO. Dichas reformulaciones reducen el problema a un solo nivel por lo que se puede utilizar un optimizador comercial y el algoritmo propuesto aplica de forma novedosa una manera de resolver el problema binivel; dicho algoritmo está basado en la metaheurística de Scatter Search y considera el equilibrio de Stackelberg durante el proceso. Además, dado que sólo encontramos un trabajo que aborda el problema binivel de la p-Mediana con preferencias de los clientes, se puede considerar que este trabajo es uno de los pioneros en analizar y resolver el p-Mediana BPO.

1.4 METODOLOGÍA

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

1. Revisión de la literatura de programación binivel.
2. Realizar una investigación de los antecedentes del problema clásico de localización de plantas.
3. Revisión de literatura del problema de localización de plantas donde se consideran las preferencias de los clientes.
4. Revisión de literatura del problema de localización de plantas con preferencias de los clientes formulado como un modelo binivel.
5. Examinar la metodología de la metaheurística Scatter Search para la resolución del problema.
6. Análisis las propiedades del problema así como el planteamiento del modelo.
7. Desarrollo e implementación computacional del algoritmo propuesto así como su evaluación computacional.
8. Se trabajó en una reformulación del problema aprovechando las relaciones primal-dual del problema del nivel inferior.
9. Durante una estancia de investigación en la Pontificia Universidad Católica de Valparaíso en Chile se llevó a cabo otra reformulación del problema donde se dejó el problema de un solo nivel sustituyendo el nivel inferior por un conjunto de restricciones.
10. Presentación de los avances de tesis en el I CSMIO, XLV SMM, XXIII ENOAN y V TLAIO/II CSMIO.
11. Comparación del desempeño de la heurística y de las reformulaciones propuestas.
12. Se sometió parte de esta tesis en una revista indexada por ISI.

1.5 ESTRUCTURA DE LA TESIS

En este primer capítulo se vio la descripción de problema de localización de plantas y se dio a conocer el objetivo central de la tesis. También se describió la justificación de la investigación y la motivación que se tuvo para realizarla. Por último se detalla la metodología propuesta para la obtención del producto final, es decir, de la tesis.

En el capítulo 2 se presenta una revisión de literatura del problema donde se presentan los trabajos que abordan el problema de la p-Mediana con preferencias ordenadas de un solo nivel y se revisa el único artículo existente en donde se consideró el p-Mediana BPO.

En el capítulo 3 se define el problema, se inicia con el planteamiento del problema abordado en esta tesis, después se presenta el modelo matemático y se termina con un análisis de la propiedad más importante del modelo donde se dan a conocer dicha característica de la naturaleza del problema que se aprovechó para las reformulaciones del problema.

En el capítulo 4 se presentan las dos reformulaciones del problema que se proponen, las cuales reducen el problema binivel a un problema de un solo nivel. En la primer reformulación se aprovechan las relaciones primal-dual del problema del seguidor y en la segunda se sustituye el nivel inferior con un conjunto de restricciones para que las preferencias de los clientes se sigan tomando en cuenta en el problema. Y por último, se presenta un análisis de las complejidades de ambas reformulaciones.

En el capítulo 5 se detalla la descripción del algoritmo que proponemos describiendo cada método que compone al algoritmo del Scatter Search y donde se considera el equilibrio de Stackelberg.

En el capítulo 6 se presentan los resultados obtenidos de la experimentación computacional, se describen los escenarios de los experimentos computacionales así

como el entorno computacional. Por último, se presentan los resultados de las reformulaciones y del algoritmo propuesto.

En el capítulo 7 se presentan las conclusiones realizadas producto de la investigación hecha en esta tesis y se presentan algunas posibles extensiones de trabajo futuro.

CAPÍTULO 2

REVISIÓN DE LITERATURA

En este capítulo se presentan los diferentes enfoques con los cuales ha sido abordado el problema de localización de plantas con preferencias. Primero se mencionan los trabajos donde el problema se resuelve como un modelo de un solo nivel y después mencionamos los trabajos encontrados donde el problema se considera como un modelo binivel. Por último, mencionamos el artículo donde trabajan el problema binivel pero con un número fijo de plantas por abrir, es decir, el problema de la *p*-Mediana, el cual está estrechamente ligado con el problema estudiado en esta tesis.

2.1 EL PROBLEMA DE LOCALIZACIÓN DE PLANTAS CON PREFERENCIAS

Hanjoul y Peeters [8] fueron los primeros en considerar las preferencias ordenadas de los clientes en el problema de localización de plantas (SPLP por sus siglas en inglés) donde suponían la existencia de una sola planta y agregaron restricciones para que se consideraran dichas preferencias, presentan además diferentes maneras de agregar éste conjunto de restricciones. Este problema se conoce como el problema simple de localización de plantas con orden (SPLPO). Para su resolución llevaron a cabo pruebas numéricas con un algoritmo heurístico voraz

basado en branch & bound. Las instancias que generaron eran pequeñas y de un tamaño fijo, consideraron problemas con 5 plantas y 8 clientes.

El problema de localización de plantas con preferencias de los clientes, así como todas sus generalizaciones está catalogado como NP-Hard ya que en Ausiello et al. [9] presentan un resultado que permite hacer esta clasificación. En el caso que las preferencias de los clientes corresponden a los costos de transporte del líder, el problema binivel de localización de plantas sin capacidades con preferencias de los clientes es equivalente al problema de localización de plantas sin capacidades. Más aún, si no existen costos fijos para abrir las plantas el problema binivel de localización de plantas sin capacidades con preferencias de los clientes sigue siendo NP-Hard. En éste artículo consideran a las preferencias de los clientes diferentes a los costos de distribución y muestran que este problema no se puede resolver en tiempo polinomial.

Después, en Cánovas et al. [10] consideran la formulación del SPLPO propuesta en Hanjoul y Peeters [8] y resuelven dos relajaciones, una para el problema entero y otra relajación fortalecida con un pre-procesamiento que solucionan utilizando el algoritmo branch and bound. Trabajan con la formulación del SPLP junto con la familia de desigualdades que proponen para considerar las preferencias de los clientes. En dichas desigualdades consideran un subconjunto de clientes y una planta a la vez, si la planta está abierta entonces los clientes no serán atendidos por una planta peor que la que está abierta de acuerdo a sus respectivas preferencias. En caso contrario, si la planta está cerrada los clientes serán atendidos por una diferente. Una vez que uno a uno los clientes seleccionan éstas plantas, un cliente no puede ser atendido por una planta que sea de sus menos preferidas puesto que indicó cuál es la más preferida de las plantas abiertas. Crean instancias tomando conjuntos de datos de una librería y generan las preferencias de los clientes con un método aleatorio de distribuciones triangulares basándose en los costos de distribución; clasifican además las instancias en tres tamaños: chicas (50 plantas y 50 clientes), mediadas (50 plantas y 75 clientes) y grandes (75 plantas y 100 clientes). Gracias al pre-procesamiento que aplican logran reducir la holgura de integralidad en tiempos razonables.

En los trabajos anteriores, el SPLPO es considerado como un problema de un solo nivel en donde se agregan restricciones y variables para poder tomar en cuenta las preferencias de los clientes como un nuevo conjunto de restricciones. Ésta es una manera de modelar el problema pero existen otras, una de ellas es formulándolo como un modelo binivel.

2.2 EL PROBLEMA BINIVEL DE LOCALIZACIÓN DE PLANTAS CON PREFERENCIAS

El primer modelo binivel del SPLPO fue propuesto en Hansen et al. [11], el modelo binivel lo reformulan como un problema de optimización de un solo nivel mediante el uso de funciones pseudo-boleanas con el propósito de relajar el problema del nivel inferior y obtener cotas inferiores válidas. Consideran supuestos adicionales: la solución óptima del seguidor es única para cualquier solución arbitraria del líder y todos los valores de las preferencias de los clientes son diferentes. Muestran que su reformulación domina tres reformulaciones (propuestas previamente) desde el punto de vista de relajación de programación lineal pero es peor que utilizando el algoritmo del problema de selección de fila de un par de matrices. Generan diez instancias al azar y suponen que las preferencias son iguales a los costos de distribución pero les introducen perturbaciones aleatorias, deciden mantener fijo el número de plantas y clientes donde ambos son de tamaño 30.

En Vasil´ev et al. [12] también reformulan el problema binivel a un modelo de uno solo utilizando desigualdades válidas relacionadas con el problema de un par de matrices. En lugar de aumentar el número de variables utilizan una nueva familia de desigualdades válidas con la cual obtienen un límite inferior que no es peor que el obtenido en Hansen et al. [11]. Muestran además que el caso cooperativo y no cooperativo puede reducirse a un único caso en el que cada cliente tiene una preferencia ordenada del conjunto de plantas que se abrirán, la solución óptima del nivel inferior es única al ser todas las preferencias diferentes entre sí por ser

ordenadas. También analizan la relación del problema con el problema de empaquetamiento usando desigualdades cliqué. Muestran que la cota inferior basada en la nueva familia de desigualdades y en los cortes del problema de empaquetamiento domina las otras cotas inferiores (propuestas previamente) desde el punto de vista de relajación lineal.

Vasil´ev y Klimentova [13] analizan la eficiencia de la familia de desigualdades que se proponen en Vasil´ev et al. [12]. Consideran una formulación combinatoria del problema y lo escriben como un problema lineal entero que es equivalente al problema binivel, para esto definen dos conjuntos de plantas: un conjunto de plantas que son mejores para un cliente que alguna planta determinada y otro conjunto que son peores para ese cliente que esa planta determinada. Ambos conjuntos son considerados en la desigualdad introducida para tomar en cuenta las preferencias de los clientes en el problema de un nivel. Implementan el método de planos cortantes basado en esa familia de desigualdades válidas y obtienen cotas inferiores. Además diseñan un algoritmo heurístico basado en recocido simulado para obtener valores cercanos al óptimo, dichos valores los utilizan como cotas superiores del problema. Con estas dos cotas obtenidas pre-procesan un algoritmo uniendo los métodos anteriores, planos cortantes y recocido simulado, mediante branch & cut y las utilizan para encontrar el óptimo. Trabajan con las instancias de Cánovas et al. [10] y presentan resultados numéricos donde muestran una reducción tanto en la holgura de optimalidad como en los tiempos obtenidos comparados con los presentados en Cánovas et al. [10]; por ejemplo, en las instancias de tamaño mediano sin utilizar la cota superior mejoran el tiempo en un 30% en promedio.

Marić et al. [14] muestran una comparación entre tres métodos metaheurísticos diseñados para resolver el SPLPO y resuelven la versión binivel del modelo. Los algoritmos propuestos son: optimización de enjambre de partículas, recocido simulado y un combinación de reducción y del método de búsqueda de vecindades variables. Resuelven el problema del seguidor con la matriz ordenada de preferencias de los clientes que se describe detalladamente en el capítulo 5 de esta tesis debido a que también recurrimos a esta técnica para conocer la respuesta del

nivel inferior. Modifican instancias de la literatura y trabajan hasta con 2000 clientes y 2000 plantas. Encuentran las soluciones óptimas de las instancias mediante CPLEX sin embargo, éste optimizador sólo fue capaz de resolver instancias con máximo 50 clientes y 50 plantas. Estos algoritmos muestran un buen rendimiento para resolver la versión binivel del problema incluso en las instancias de grandes dimensiones, no obstante el método que mostró un mejor rendimiento en cuestión de soluciones de calidad y tiempo computacional fue el algoritmo híbrido.

Camacho-Vallejo et al. [15] proponen dos reformulaciones del problema binivel reduciéndolo a un problema de programación entera mixta de un solo nivel mediante el uso de las relaciones primal-dual del nivel inferior. Además, debido a la dificultad del problema utilizan un algoritmo evolutivo basado en el equilibrio de Stackelberg para resolver el problema formulado como un modelo binivel. Para probar el rendimiento del algoritmo y de las reformulaciones trabajan con las instancias de Cánovas et al. [10] y generan otro conjunto de instancias de mayor tamaño bajo el mismo procedimiento empleado en Cánovas et al. [10], obtienen instancias hasta de 500 plantas y 1000 clientes. Ambas reformulaciones que proponen las resuelven de manera exacta con un software de optimización pero sólo en las instancias pequeñas encuentran la solución óptima, por este motivo proponen un procedimiento heurístico en donde resuelven el nivel inferior de manera exacta utilizando el mismo procedimiento de Marić et al. [14]. Una de las reformulaciones tarda más de 40 minutos en resolver de manera óptima las instancias de 50 plantas y de 50 clientes mientras que la otra reformulación se demora más de 80 minutos en resolverlas. Sólo una de las reformulaciones logra resolver las instancias de 50x75 pero en tiempos no muy aceptables por tal motivo justifican la resolución del problema mediante el algoritmo Stackelberg-Evolutivo que proponen. Los resultados de la experimentación computacional de éste algoritmo muestran que tiene un buen desempeño ya que en más de la mitad de las veces que lo ejecutaron encuentra el valor óptimo y en el resto las holguras de optimalidad son muy pequeñas, además los tiempos son razonables con un tiempo máximo de 2, 53 y 250 segundos para las instancias chicas, medianas y grandes respectivamente.

2.3 EL PROBLEMA BINIVEL DE LA p-MEDIANA CON PREFERENCIAS

Después de una intensa búsqueda sólo encontramos un artículo en donde se presenta el SPLPO con un modelo binivel en el cual se considera un número fijo de plantas por abrir. Este trabajo fue escrito por Alekseeva y Kochetov [16] en donde proponen varias reformulaciones basadas en el problema de un par de matrices reescribiendo el problema como uno de un sólo nivel. Diseñan un algoritmo genético híbrido para resolver las reformulaciones donde consideran cuatro tipos de cruza (uniforme, voraz, de un solo punto y reencadenamiento de trayectorias) y procedimientos de búsqueda local como mutación (heurística Lin-Kernighan). Validan el desempeño de su algoritmo utilizando instancias de 100 plantas y 100 clientes. Utilizan un software comercial para encontrar la solución óptima. Después de 270 horas de trabajo las reformulaciones no alcanzan la solución óptima, mientras que el algoritmo genético híbrido encuentra mejores soluciones en 2 minutos en el mismo equipo.

El trabajo de Alekseeva y Kochetov [16] difiere principalmente con lo desarrollado en esta tesis por el hecho de que ellos no consideran el costo de instalación de las plantas, es decir, ellos solamente consideran el costo de distribución asociado a la asignación planta-cliente. Además, es importante mencionar que la heurística que proponen soluciona una reformulación del problema, es decir, el problema no lo consideran como un modelo de optimización binivel.

Nosotros proponemos dos reformulaciones del p-Mediana BPO donde lo reducimos a un problema de un sólo nivel empleando una propiedad del modelo que se describe en el siguiente capítulo, ambas reformulaciones se realizan considerando el problema modelado mediante programación binivel. Debido a la complejidad de problema y al tiempo computacional empleado en las dos reformulaciones proponemos además la implementación de una metaheurística para resolver el problema considerándolo como un problema de programación binivel. Cabe recordar que también se trata de un problema combinatorio difícil y que también, un

subconjunto de variables se determinan implícitamente al resolver otro problema de programación matemática, lo cual lo hace aún más difícil de resolver.

Nuestro algoritmo se basa en el equilibrio de Stackelberg [17] que refleja el caso de la competencia en un mercado compuesto por un líder que desea minimizar los costos de instalación y de distribución, y un seguidor que quiere optimizar las preferencias de los clientes. Durante el proceso iterativo de construcción de soluciones del problema binivel, el hecho de considerar la solución óptima del seguidor para cada solución del líder refleja la intención de encontrar el equilibrio de Stackelberg. Dicho equilibrio se encuentra cuando el líder selecciona la mejor decisión para él tomando en cuenta la solución óptima del seguidor.

Además proponemos el uso de la metaheurística Scatter Search debido a su versatilidad y buen rendimiento para resolver problemas complejos. Scatter Search es un algoritmo poblacional introducido en los años setenta, tiene algunas similitudes con los algoritmos genéticos. Las principales diferencias radican en que Scatter Search utiliza estrategias sistemáticas en lugar de aleatorias y que utiliza un número menor de soluciones para conformar la población. Además pertenece a la familia de los llamados algoritmos evolutivos que trabajan con una población de individuos que representan las soluciones de un problema a resolver. Esta población se modifica por operadores genéticos y después se selecciona a los individuos que sobreviven para que formen la siguiente generación. Después de un número de generaciones se espera que los individuos mejoren y se parezcan a la solución que se busca debido a que éstos métodos están basados en la evolución biológica por lo que se espera que sobrevivan los más aptos. Una descripción detallada de Scatter Search y varias de sus aplicaciones se puede encontrar en Laguna y Martí [18].

CAPÍTULO 3

El p-Mediana BPO

En este capítulo se describe detalladamente el problema estudiado en esta tesis así como el modelo, la formulación matemática, los supuestos considerados y la propiedad más importante del modelo. Se inicia este capítulo con una introducción sobre la programación binivel y sus conceptos principales.

3.1 PROGRAMACIÓN BINIVEL

Como mencionamos en el capítulo anterior, Heinrich von Stackelberg [17] usó por primera vez un modelo jerárquico para describir las situaciones del mercado real en 1934. En ese modelo diferentes agentes pretenden tomar la mejor decisión en el mercado con respecto a ellos mismos, pero no pueden tomar sus decisiones de manera independiente porque se ven forzados a actuar de acuerdo a una jerarquía existente en el mercado. Si sólo existen dos agentes (caso más simple) entonces esta jerarquía tiene dos niveles, un líder que toma las decisiones de manera independiente y un seguidor que toma sus decisiones considerando la decisión del líder.

Si únicamente existen dos niveles entonces estos problemas pertenecen al área de programación binivel. De forma general un problema puede dividirse en dos vectores de variables: x y y. El líder toma la decisión y mientras que el seguidor considera esa decisión y como parámetro y optimiza su problema regresando la mejor decisión para él sabiendo que la y está fija. El problema consiste en encontrar la

decisión y que al considerar la $x^*(y)$ minimice su función objetivo. El problema que tiene que resolverse es el llamado Juego de Stackelberg no cooperativo, donde cada jugador quiere maximizar sus ganancias. Además, se dice que es un juego de información completa puesto que el líder es el primero en tomar una decisión y el seguidor toma una decisión conociendo de antemano el movimiento que realizó el líder. A continuación se presenta la formulación de éste problema.

Sean X y Y los conjuntos de estrategias del seguidor y del líder respectivamente y sean las funciones de ganancia del líder y del seguidor $f_L(x,y)$ y $f_F(x,y)$ respectivamente. Entonces, el seguidor toma una decisión conociendo la decisión y del líder, es decir, el seguidor elegirá su mejor estrategia $x^*(y)$ de tal manera que maximice su función de ganancia en X:

$$x(y) \in \psi(y) \coloneqq Arg\ max_x\{f_F(x,y): x \in X\} \tag{3.1}$$

A su vez, el líder tomará su mejor decisión en Y:

$$"max_y"\{f_L(x,y): y \in Y, x \in \psi(y)\} \tag{3.2}$$

Si existe más de un jugador en un mismo nivel, es decir, si hay más de un líder o más de un seguidor entonces los jugadores deben buscar un equilibrio entre ellos, el equilibrio de Nash.

Un modelo general de un problema de programación binivel se presenta en Dempe [19] y es el que se escribe a continuación. Sean $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $a \in \mathbb{R}^p$y las matrices A^1 y A^2 de dimensiones apropiadas así como el vector c. El problema del nivel inferior quedaría:

$$min_x\{\langle c,x\rangle: A^1x \leq a - A^2y, x \geq 0\} \tag{3.3}$$

Por lo que el conjunto de soluciones óptimas del seguidor sería:

$$\psi_L(y) = Arg\ min_x\{\langle c,x\rangle: A^1x \leq a - A^2y, x \geq 0\} \tag{3.4}$$

Así, el problema binivel puede definirse como:

$$"min_y"\{\langle d^1, x\rangle + \langle d^2, y\rangle : A^3 y = b, y \geq 0, x \in \psi_L(y)\} \tag{3.5}$$

donde $b \in \mathbb{R}^l$es un vector de dimensiones apropiadas.

Puede existir el caso donde la solución del problema del seguidor no sea única, para denotar la existencia de esa incertidumbre se escribe entre comillas el objetivo del líder, es decir $"min_y"$.

Como el líder no tiene control sobre la elección del seguidor entonces existen varias situaciones que pueden presentarse dependiendo del nivel de cooperación entre los jugadores. Por ejemplo, si el seguidor tiene dos decisiones iguales para él pero que a su vez representan diferentes valores de la función objetivo del líder, es claro que el líder no puede influir en la decisión del seguidor para que éste tome la más conveniente para él, si asumimos que el seguidor toma la decisión favorable para el líder entonces este caso se conoce como el problema de optimización optimista:

$$min_{x,y}\{\langle d^1, x\rangle + \langle d^2, y\rangle : A^3 y = b, y \geq 0, x \in \psi_L(y)\} \tag{3.6}$$

Si éste problema tiene solución óptima se puede definir de la siguiente manera:

$$min_y\{\varphi_0(y) + \langle d^2, y\rangle : A^3 y = b, y \geq 0\} \tag{3.7}$$

donde
$$\varphi_0(y) = min_x\{\langle d^1, x\rangle : x \in \psi_L(y)\} \tag{3.8}$$

En el caso contrario, cuando el líder no puede influir en la decisión del seguidor, sólo se puede limitar a minimizar el impacto que le puede ocasionar una decisión del seguidor que no le resulte benéfica. Es decir, cada jugador (líder y seguidor) busca un beneficio propio sin importarle los demás participantes. El problema de optimización pesimista es el siguiente:

$$min_y\{\varphi_p(y) + \langle d^2, y\rangle : A^3 y = b, y \geq 0\} \tag{3.9}$$

donde
$$\varphi_p(y) = max_x\{\langle d^1, x\rangle : x \in \psi_L(y)\} \tag{3.10}$$

Todas las ecuaciones presentadas anteriormente así como la descripción de cada una de ellas pueden verse en Dempe [19]. Sin embargo, en Bard [20] presentan otra manera de definir un problema general de programación binivel y es el que se presenta a continuación:

$$\min_{x \in X} F(x, y) \tag{3.11}$$

sujeto a: $G(x, y) \leq 0$ (3.12)

$$\min_{y \in Y} f(x, y) \tag{3.13}$$

sujeto a: $g(x, y) \leq 0$ (3.14)

$x, y \geq 0$ (3.15)

donde el problema del líder está definido por (3.11)-(3.13), (3.1) es la función objetivo del líder que desea minimizar la función $F(x, y)$, dicho problema tiene dos conjuntos de restricciones: (3.12) y (3.13) donde la última restricción es otro problema de programación matemática. Este problema es el problema del seguidor que está definido por las ecuaciones (3.13)-(3.15), la función objetivo del seguidor es (3.13) donde quiere minimizar la función $f(x, y)$ bajo un par conjuntos de restricciones: (3.14) y (3.15) donde ésta última es la restricción de signo de las variables de decisión del problema.

Partiendo del modelo anterior se pueden mencionar unas definiciones que se presentan a continuación debido a que también son de suma importancia en programación binivel:

1. Región de restricciones del problema de programación binivel:

$$S = \{(x, y): x \in X, y \in Y, G(x, y) \leq 0, g(x, y) \leq 0\} \tag{3.16}$$

2. Conjunto factible del seguidor, para cada $x \in X$:

$$S(x) = \{y \in Y: g(x, y) \leq 0\} \tag{3.17}$$

3. Conjunto de reacciones racionales del seguidor, para cada $x \in X$:

$$P(x) = \{y \in Y: y \in Arg\, min\{f(x, y): y \in S(x)\}\} \tag{3.18}$$

4. Región inducible:

$$IR = \{(x, y): (x, y) \in S, y \in P(x)\} \tag{3.19}$$

Como el seguidor reacciona ante una decisión tomada por el líder, el conjunto factible de reacciones racionales del seguidor define dicha respuesta. Se dice que es de reacciones racionales debido a que el seguidor tomará la decisión que más le convenga, es decir, analizará la situación que más le beneficie de manera consciente. La región inducible representa el conjunto sobre el cual el líder optimizará, es decir, es el conjunto factible del líder. Puede verse que en éste conjunto la y está restringida por la reacción del seguidor.

Si el conjunto de reacciones racionales tiene más de un valor para x entonces el líder no podría alcanzar su máximo objetivo, además puede presentarse el caso donde podría ser discontinuo. La región inducible usualmente es no convexa y puede ser hasta vacía o no conexa por algunas de las restricciones del líder. También, se puede presentar el caso donde a pesar de que las funciones puedan ser continuas y acotadas no se pueda garantizar que existe solución. Por lo tanto, puede verse fácilmente que los problemas de programación binivel no son fáciles de tratar. Además, los problemas de programación binivel están clasificados como NP-Hard. También en Bard [21] presentan una demostración de un caso particular, de que el problema de programación binivel lineal es NP-Hard.

También puede presentarse el caso donde el problema de programación binivel es convexo, es decir, que el problema del seguidor sea convexo; dicho problema ha sido de interés para muchos investigadores ya que tienen una ventaja, bajo procedimientos apropiados, se puede obtener un problema equivalente de un nivel. Existen varias técnicas de reducción de los problemas de dos niveles a un problema de un solo nivel, las técnicas más comunes son: Condiciones de Karush-Kuhn-Tucker, teoremas de funciones implícitas, relación primal-dual, entre otras. No obstante, aunque un problema de programación binivel sea convexo su región

inducible puede ser no conexa e incluso vacía al considerar las restricciones del nivel superior. Las propiedades de los problemas de programación binivel y algunos algoritmos para su resolución pueden verse en Vicente y Calamai [22].

3.2 PLANTEAMIENTO DEL PROBLEMA

Como ya se mencionó anteriormente, la programación binivel ha servido para modelar problemas de logística tomando en cuenta algunos componentes involucrados en el entorno del problema en específico que antes no se consideraban de manera conjunta. Hay una gran variedad de problemas que se pueden modelar con programación binivel; por ejemplo problemas de localización, problemas de transporte, diseño de redes, logística portuaria, estudios ambientales, logística humanitaria, entre otros. Nosotros trabajamos con el problema de localización de plantas el cual se describió en el capítulo 1.

El problema analizado en esta tesis es una extensión del SPLP en donde se consideran los mismos supuestos que en Hansen et al. [11] y se toman en cuenta algunas consideraciones adicionales: se tiene un número fijo de plantas por abrir y las plantas tienen capacidad infinita.

Examinamos el p-Mediana BPO sin capacidad en las plantas formulado como un modelo binivel donde el objetivo del líder es minimizar los costos de instalación y de distribución; y el seguidor optimiza las preferencias ordenadas de los clientes. El objetivo del problema es abrir el número p de plantas que minimicen el costo total (instalación y distribución) de localización y, a su vez, minimicen las preferencias ordenadas de los clientes. Al igual que en Vasil´ev et al. [12] consideramos que la solución óptima del nivel inferior es única al ser las preferencias ordenadas y diferentes entre sí.

3.3 FORMULACIÓN MATEMÁTICA

A continuación se presenta la formulación matemática del p-Mediana BPO. Se definen los conjuntos, parámetros, variables de decisión y los supuestos considerados en el problema; también se presenta el modelo matemático y algunas de sus propiedades. El modelo binivel considerado en esta tesis es muy similar al que se propuso en Vasil´ev et al. [12], la diferencia radica en que agregamos una restricción en el nivel superior con la que se asegure abrir un número fijo de plantas ya que nosotros trabajamos con el problema de la p-Mediana.

Conjuntos

$I = \{1, 2, \dots, |I|\}$ = conjunto de plantas, donde $|I|$ es la cardinalidad del conjunto I,

$J = \{1, 2, \dots, |J|\}$ = conjunto de clientes, donde $|J|$ es la cardinalidad del conjunto J,

i = índice de las plantas; $i \in I$,

j = índice de los clientes; $j \in J$.

Parámetros

c_{ij} = costo de abastecer toda la demanda del cliente j por la planta i,

f_i = costo fijo por abrir la planta i,

g_{ij} = preferencia del cliente j de ser servido por la planta i,

p = número de plantas que deben abrirse.

Variables de decisión

Las variables de decisión del problema son dos y son de naturaleza binaria, se describen a continuación.

$y_i = \begin{cases} 1 & \text{si la planta } i \text{ se abre} \\ 0 & \text{de otro modo.} \end{cases}$

$x_{ij} = \begin{cases} 1 & \text{si la planta } i \text{ satisface la demanda del cliente } j \\ 0 & \text{de otro modo.} \end{cases}$

Supuestos

1) Los clientes, a priori, establecen sus preferencias ordenadas para cada una de las plantas en una lista de números del 1 al $|I|$, donde 1 es la planta más preferida, $|I|$ es la menos preferida y los valores de las preferencias son diferentes (para cada cliente j, $g_{ij} \neq g_{ij}$ para toda planta i). Además, dado que las preferencias son diferentes la solución del nivel inferior es única (ver Hansen et al. [11] y Vasil´ev et al. [12]).
2) Las plantas no tienen restricción de capacidad, es decir, una planta puede abastecer la demanda de varios clientes pero un cliente debe ser abastecido por una sola planta.
3) Se deben abrir exactamente el número p de plantas predefinido.

El modelo matemático del p-Mediana BPO es el siguiente:

$$\min_{y,x} \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{3.20}$$

sujeto a:

$$\sum_{i \in I} y_i = p \tag{3.21}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{3.22}$$

$$x \in \operatorname{Arg\,min} \sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} \tag{3.23}$$

sujeto a:

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J \qquad (3.24)$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J \qquad (3.25)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \qquad (3.26)$$

El problema del nivel superior está definido por (3.20)-(3.23), donde (3.20) es la función objetivo del líder que desea minimizar ambos costos de instalación y de distribución, (3.21) indica el número total p de plantas a abrir, (3.22) establece la restricción binaria para cada variable y_i y finalmente, (3.23) es la restricción que indica que las variables x_{ij} son controladas por el seguidor, están implícitamente determinadas por la solución óptima del problema del nivel inferior. Este problema está definido por las ecuaciones (3.23)-(3.26), donde la función objetivo del seguidor es (3.23) en donde se quiere minimizar las preferencias ordenadas de los clientes, en (3.24) se asegura que cada cliente sea abastecido solamente por una planta, (3.25) indica que los clientes se pueden asignar únicamente a plantas abiertas, y finalmente, (3.26) es la restricción de signo de la variable de decisión binaria x_{ij}.

3.4 ANÁLISIS DE LAS PROPIEDADES DEL MODELO

Es importante mencionar que la propiedad más importante que posee la matriz asociada a las restricciones del problema del nivel inferior es que es totalmente unimodular. Recordando la definición, una matriz es totalmente unimodular si el determinante de cada una de sus submatrices cuadradas tiene valor de +1, -1 o 0. La demostración de ésta propiedad puede verse en Bazaraa et al. [23].

Ahora bien, las dos variables de nuestro problema son binarias por lo que la variable del seguidor x_{ij} sólo puede tener el valor de +1 o 0 y ésta representa si la planta i safisface la demanda del cliente j**.** Si juntamos los dos conjuntos de restricciones del seguidor en una matriz quedaría de la siguiente manera:

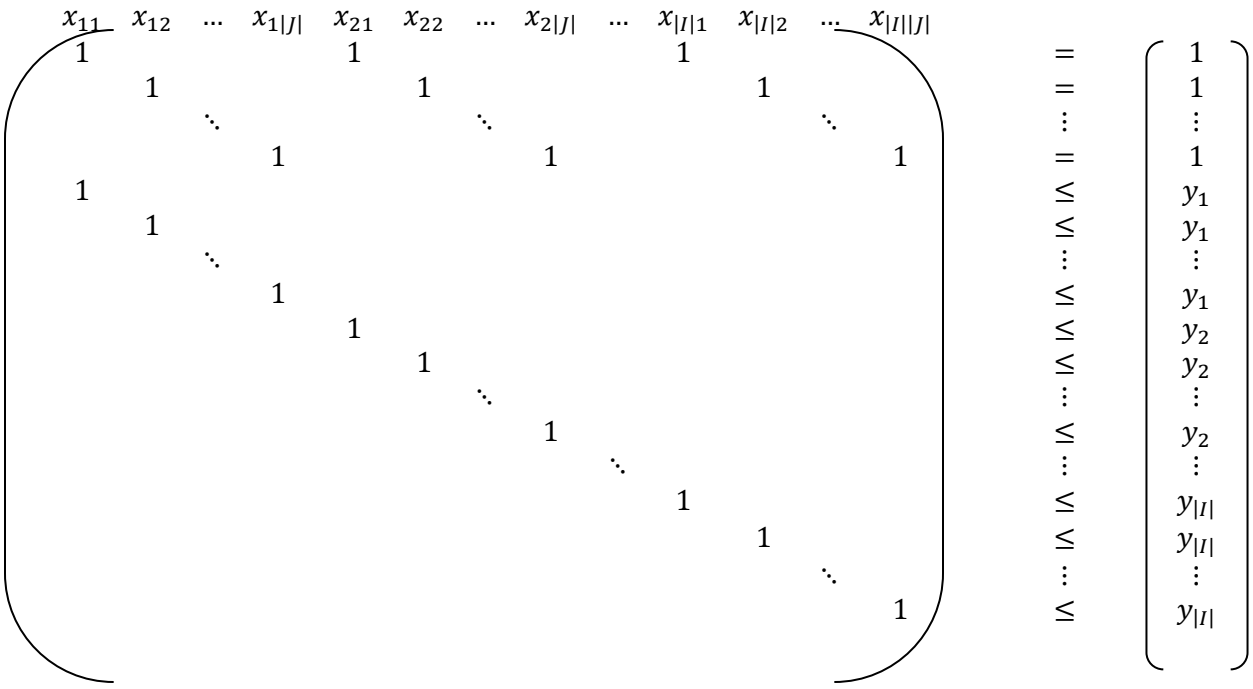

Figura 3.1 Matriz asociada a las restricciones del seguidor

Es fácil ver que los dos conjuntos de restricciones del nivel inferior (ignorando la restricción de signo) (3.24) y (3.25) formar una matriz que está compuesta únicamente por valores de 0 o 1. Por lo tanto, podemos concluir que la matriz asociada a las restricciones del seguidor es unimodular, ésta propiedad garantiza que si se resuelve el problema como un problema de programación lineal, el resultado serán variables de decisión enteras. Por consiguiente, las variables x_{ij} se pueden redefinir como variables no negativas, es decir, $x_{ij} \geq 0 \; \forall i \in I, j \in J$ sin que se vea afectada la integralidad de las soluciones del seguidor y resolver el problema de programación lineal. Ésta propiedad fue de gran utilidad en una de las reformulaciones que se presentan en el siguiente capítulo porque nos permitió relajar el nivel inferior sin afectar la restricción binaria de las variables x_{ij} convirtiendo el problema en un problema lineal.

CAPÍTULO 4

REFORMULACIONES DEL PROBLEMA

En esta sección se describen dos reformulaciones del p-Mediana BPO. La primera reformulación se obtiene al hacer uso de las relaciones primal-dual del problema del nivel inferior. Por otro lado, al analizar la complejidad del modelo resultante al reducir el problema binivel a un problema de un solo nivel nos dimos cuenta que los modelos siguen siendo complejos. Es por esto que se pensó una forma alternativa de escribir el modelo matemático de tal forma que las preferencias de los clientes se sigan considerando sin tener propiamente un problema binivel. Es importante recordar que un subconjunto de las variables están implicitamente determinadas por la solución de otro problema de optimización por lo que la dificultad para resolver el problema aumenta de sobremanera. Este hecho nos motivó a pensar en una forma alternativa de reformularlo y resolverlo de manera exacta.

4.1 REFORMULACIÓN 1: USANDO LAS RELACIONES PRIMAL-DUAL DEL NIVEL INFERIOR

En esta primera reformulación se utilizan las condiciones de optimalidad del problema del nivel inferior y su correspondiente problema dual. Si las variables y son consideradas como fijas, entonces el problema del nivel inferior puede ser visto como un problema de asignación. Este problema de asignación tiene la particularidad de que la matriz de restricciones del problema del seguidor es unimodular, por lo tanto

las restricciones de binariedad para las variables x_{ij} se reemplazan por restricciones de no negatividad, es decir, $x_{ij} \geq 0, \ \forall i \in I, j \in J$. Al tener ahora un programa lineal en el problema del nivel inferior se pueden obtener las condiciones de optimalidad primal-dual. Entonces, para reducir el problema binivel a un problema de un solo nivel se pueden usar estas relaciones primal-dual garantizando la optimalidad del problema binivel.

Sean $\alpha_j, \forall j \in J$ y $\beta_{ij}, \forall i \in I, j \in J$ las variables duales asociadas a las restricciones del problema del seguidor, donde α_j es la variable dual para la restricción (4.4) del primal mientras que la variable dual β_{ij} es para la restricción (4.5). Entonces el problema dual del nivel inferior queda como sigue:

$$\min_{y,x,\alpha,\beta} \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{4.1}$$

sujeto a:

$$\sum_{i\in I} y_i = p \tag{4.2}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{4.3}$$

$$\sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{4.4}$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J \tag{4.5}$$

$$\alpha_j + \beta_{ij} \leq g_{ij} \qquad \forall i \in I, j \in J \tag{4.6}$$

$$\sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} = \sum_{j\in J}\alpha_j + \sum_{i\in I}\sum_{i\in J}\beta_{ij}y_i \tag{4.7}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{4.8}$$

$$\beta_{ij} \leq 0 \qquad \forall i \in I, j \in J \tag{4.9}$$

donde el problema descrito en (4.1)-(4.9) se obtiene considerando la variable del líder y_i como parámetro para el problema del nivel inferior. Las restricciones (4.2)-(4.5) y (4.8) aseguran la factibilidad primal, las restricciones (4.6) y (4.9) aseguran la factibilidad dual, y la restricción (4.7) garantiza que los valores de la función objetivo del seguidor que corresponden a sus problemas primal y dual son iguales. Como conclusión se puede establecer que éste problema es equivalente al p-Mediana BPO, sin embargo a pesar de que el problema reformulado es un problema de un solo nivel es difícil de resolver debido al termino cuadrático de la restricción (4.7) que aparece como resultado del hecho de que la variable y_i es un parámetro para el nivel inferior pero en la reformulación es considerada como una variable. Además el

problema (4.1)-(4.9) agrega $|I||J| + |J|$ variables y $|I||J| + 1$ restricciones donde uno de los términos es no lineal.

Con el fin de resolver éste problema, la restricción (4.7) necesita ser linealizada. Las variables $\pi_{ij} = \beta_{ij} y_i$ se introducen asegurando que cuando $y_i = 0$, entonces $\pi_{ij} = 0$; y si $y_i = 1$, entonces $\pi_{ij} = \beta_{ij}$. Esto se puede hacer mediante la introducción de las siguientes desigualdades:

$$\pi_{ij} \leq 0 \qquad \forall i \in I, j \in J \qquad (4.10)$$
$$\pi_{ij} \geq -M y_i \qquad \forall i \in I, j \in J \qquad (4.11)$$
$$\pi_{ij} \geq \beta_{ij} \qquad \forall i \in I, j \in J \qquad (4.12)$$
$$\pi_{ij} + M y_i \leq \beta_{ij} + M \qquad \forall i \in I, j \in J \qquad (4.13)$$

Reemplazamos la ecuación (4.7) por la siguiente ecuación:

$$\sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} = \sum_{j \in J} \alpha_j + \sum_{i \in I} \sum_{i \in J} \pi_{ij} \qquad (4.14)$$

Como resultado, se genera un modelo de programación entera mixta que es equivalente al p-Mediana BPO. El modelo reformulado queda de la siguiente manera:

$$\min_{y,x,\alpha,\beta} \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \qquad (4.1)$$

sujeto a:

$$\sum_{i \in I} y_i = p \qquad (4.2)$$
$$y_i \in \{0,1\} \qquad \forall i \in I \qquad (4.3)$$
$$\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J \qquad (4.4)$$
$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J \qquad (4.5)$$
$$\alpha_j + \beta_{ij} \leq g_{ij} \qquad \forall i \in I, j \in J \qquad (4.6)$$
$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \qquad (4.8)$$
$$\beta_{ij} \leq 0 \qquad \forall i \in I, j \in J \qquad (4.9)$$
$$\pi_{ij} \leq 0 \qquad \forall i \in I, j \in J \qquad (4.10)$$
$$\pi_{ij} \geq -M y_i \qquad \forall i \in I, j \in J \qquad (4.11)$$
$$\pi_{ij} \geq \beta_{ij} \qquad \forall i \in I, j \in J \qquad (4.12)$$
$$\pi_{ij} + M y_i \leq \beta_{ij} + M \qquad \forall i \in I, j \in J \qquad (4.13)$$
$$\sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} = \sum_{j \in J} \alpha_j + \sum_{i \in I} \sum_{i \in J} \pi_{ij} \qquad (4.14)$$

Cabe mencionar que ésta reformulación del problema, utilizando las relaciones primal-dual, es sólo una alternativa para resolverlo. Existe otra forma de reformular el problema binivel y es utilizando la holgura complementaria, para esta alternativa la restricción (4.7) que es la igualdad de las funciones objetivos de ambos problemas puede ser sustituida por las siguientes restricciones de complementariedad que fuerzan la holgura complementaria:

$$x_{ij}(\alpha_j + \beta_{ij} - g_{ij}) = 0 \qquad \forall i \in I, j \in J \qquad (4.15)$$
$$\beta_{ij}(x_{ij} - y_i) = 0 \qquad \forall i \in I, j \in J \qquad (4.16)$$

Es fácil ver que ambas restricciones son no lineales, sin embargo pueden ser linealizadas con las siguientes expresiones:

$$\alpha_j + \beta_{ij} - g_{ij} \geq -\mathrm{M}(1 - x_{ij}) \qquad \forall i \in I, j \in J \qquad (4.17)$$
$$\beta_{ij} \geq -M\left(1 + (x_{ij} - y_i)\right) \qquad \forall i \in I, j \in J \qquad (4.18)$$

Entonces el problema resultante de programación entera mixta es equivalente al p-Mediana BPO y puede ser considerado como la segunda linealización a un nivel del problema. El modelo reformulado queda de la siguiente manera:

$$\min_{y,x,\alpha,\beta} \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \qquad (4.1)$$

sujeto a:

$$\sum_{i\in I} y_i = p \qquad (4.2)$$
$$y_i \in \{0,1\} \qquad \forall i \in I \qquad (4.3)$$
$$\sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \qquad (4.4)$$
$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J \qquad (4.5)$$
$$\alpha_j + \beta_{ij} \leq g_{ij} \qquad \forall i \in I, j \in J \qquad (4.6)$$
$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \qquad (4.8)$$
$$\beta_{ij} \leq 0 \qquad \forall i \in I, j \in J \qquad (4.9)$$
$$\alpha_j + \beta_{ij} - g_{ij} \geq -\mathrm{M}(1 - x_{ij}) \qquad \forall i \in I, j \in J \qquad (4.17)$$
$$\beta_{ij} \geq -M\left(1 + (x_{ij} - y_i)\right) \qquad \forall i \in I, j \in J \qquad (4.18)$$

Analizando los dos modelos resultantes de la reformulación del p-Mediana BPO nos dimos cuenta que este último modelo es mucho menor en tamaño que el de la igualdad de las funciones objetivo. Por lo tanto, decidimos trabajar únicamente con el de la holgura complementaria definido por (4.1)-(4.6), (4.8)-(4.9) y (4.17)-(4.18).

4.2 REFORMULACIÓN 2: SUSTITUCIÓN DEL NIVEL INFERIOR POR UN CONJUNTO DE RESTRICCIONES EQUIVALENTES

Al observar el número de restricciones y de variables consideradas en la reformulación anterior definidas por (4.1)-(4.6), (4.8)-(4.9) y (4.17)-(4.18) nos dimos cuenta que eran considerablemente grandes. Es por esto que decidimos investigar una manera alternativa donde las preferencias de los clientes aún se consideraran (el problema del nivel inferior). Proponemos otra reformulación del problema donde lo reducimos nuevamente a un problema de un solo nivel en la cual sustituimos el problema del seguidor por el siguiente conjunto de restricciones.

$$\sum_{i\epsilon I} x_{ij} g_{ij} \leq g_{ij}\, y_i + G_{max}(1 - y_i) \qquad \forall i \in I, j \in J \qquad (4.19)$$

donde $$G_{max} = max\left(g_{ij}\right) + 1 \quad \forall i\epsilon I, j\epsilon J \qquad (4.20)$$

Es fácil ver que el problema del seguidor se considera en (4.19) donde el problema sigue tomando en cuenta las preferencias ordenadas de los clientes. En detalle, si la planta i se abre, entonces la expresión $G_{max}(1 - y_i)$ se hace cero y la restricción (4.19) se satisface por (4.4) y (4.5). En caso contrario, si la planta i no se abre, la expresión $g_{ij}y_i$ se hace cero y la suma de las preferencias seleccionadas por los clientes (lado izquierdo de la desigualdad) será siempre menor que G_{max}, esto es, la restricción (4.19) será relajada ya que G_{max} es una cota superior para todas las preferencias ordenadas.

El problema propuesto es equivalente al problema binivel analizado en esta tesis y se muestra a continuación:

$$\min_{y,x} \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{4.1}$$

sujeto a:

$$\sum_{i\in I} y_i = p \tag{4.2}$$

$$y_i \in \{0,1\} \tag{4.3}$$

$$\sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{4.4}$$

$$x_{ij} \le y_i \qquad \forall i \in I, j \in J \tag{4.5}$$

$$\sum_{i\epsilon I} x_{ij} g_{ij} \le g_{ij}\, y_i + G_{max}(1 - y_i) \qquad \forall i \in I, j \in J \tag{4.19}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{4.8}$$

4.3 ANÁLISIS DE LAS COMPLEJIDADES DE LAS REFORMULACIONES

Es importante mencionar que si el problema se formula como un modelo binivel, tendrá sólo dos restricciones (sin considerar las restricciones de signo), donde una de ellas es otro problema de optimización. Comparándo ambas reformulaciones propuestas, se puede ver que en la segunda el número de variables se mantiene pero se incrementa el número de restricciones, en este caso, tiene $2|I||J| + |J| + 1$ restricciones en total; sin embargo la primer reformulación tiene más variables y restricciones cotejándola con la segunda.

Ambos enfoques del modelo tienen su grado de dificultad, en el modelo binivel hay un problema de optimización dentro de una restricción lo cual complica su resolución; por el otro lado, en el modelo reformulado aumenta el número de restricciones y de variables.

Sin embargo, ambas reformulaciones propuestas del problema a pesar de aumentar de tamaño pueden resolverse mediante un software especializado para

optimizar los problemas de programación matemática. Por otra parte, a nuestro conocimiento, aún no existe un software comercial capaz de resolver un problema binivel de forma general. Basándonos en este hecho, en el siguiente capítulo proponemos un algoritmo heurístico para encontrar soluciones de buena calidad en tiempo razonable para el p-Mediana BPO donde no reformulamos el problema sino que trabajamos directamente sobre el modelo binivel.

CAPÍTULO 5

ALGORITMO STACKELBERG-SCATTER SEARCH

En esta sección se describe a detalle el algoritmo que proponemos en esta tesis. Como vimos en el capítulo 2, nuestro algoritmo se basa en la metaheurística Scatter Search y en el clásico juego de Stackelberg no cooperativo. La metaheurística inicia cuando el líder toma una decisión y y el seguidor considera esa decisión como parámetro y optimiza su problema regresando la mejor decisión $x^*(y)$ para él sabiendo que la y está fija, a su vez el líder considera esa solución $x^*(y)$ y optimiza su problema; esta interacción entre ambos niveles finaliza hasta que se llega al equilibrio de Stackelberg.

5.1 DESCRIPCIÓN DEL ALGORITMO

Durante la implementación del algoritmo propuesto se llevó a cabo el siguente esquema: una vez que en el nivel superior se contruye una solución entonces se resuelve el problema del nivel inferior de forma exacta. Como ya habíamos mencionado en el capítulo 2, debido al alto costo computacional de resolver de manera exacta con un optimizador el problema del seguidor decidimos recurrir a la metodología de la matriz ordenada de las preferencias de los clientes presentada en Marić et al. [14], la cual se explica a continuación.

En un arreglo se ordenan las plantas potenciales de acuerdo a las preferencias de cada cliente j, la más preferida va al inicio y la menos preferida al final. Los arreglos

de todos los clientes se guardan en una matriz llamada matriz ordenada de preferencias. Después, para cada cliente $j \in J$ se analiza si su planta más preferida está abierta, si es así se asigna ese cliente a esa planta, en caso contrario se procede con la segunda más preferida y se vuelve a analizar si está abierta dicha planta. Si no es así, se repite el proceso hasta que se asigne ese cliente j a la planta abierta i más preferida por el cliente. De esta manera se obtiene la matriz de asignación de los clientes a las plantas.

Un algoritmo básico del Scatter Search consta de cinco métodos, los cuales son:

1. Método generador de diversificación.
2. Método de creación y actualización del conjunto referencia.
3. Método generador del subconjunto del conjunto referencia.
4. Método de combinación.
5. Método de mejora.

La metaheurística empieza creando un conjunto de soluciones diversas, llamado $Población$, de tamaño P_{size} elementos de los cuales se extraen b elementos para formar el conjunto referencia ($RefSet$) que debe contener $^{b}/_{2}$ soluciones de calidad y $^{b}/_{2}$ soluciones diversas.

Se crea un conjunto donde se introducen todas las posibles parejas de soluciones combinadas del conjunto referencia, una vez creada la nueva solución pasa al método de mejora, el cual se divide en dos fases:

1. Fase de factibilidad: la solución se vuelve factible si no lo era. El método de combinación no garantiza que las nuevas soluciones creadas sean factibles.
2. Fase de mejora: la solución se intenta mejorar mediante una búsqueda local.

Las soluciones combinadas pueden entrar al conjunto referencia y reemplazar a algunas que se encuentran dentro siempre y cuando sean mejores que ellas en base al valor de la función objetivo. El algoritmo se detiene cuando se llega a un criterio de

paro, en este caso cuando ya no existen nuevas soluciones que combinar del $RefSet$. En la Figura 5.1 se muestra una ilustración del Scatter Search.

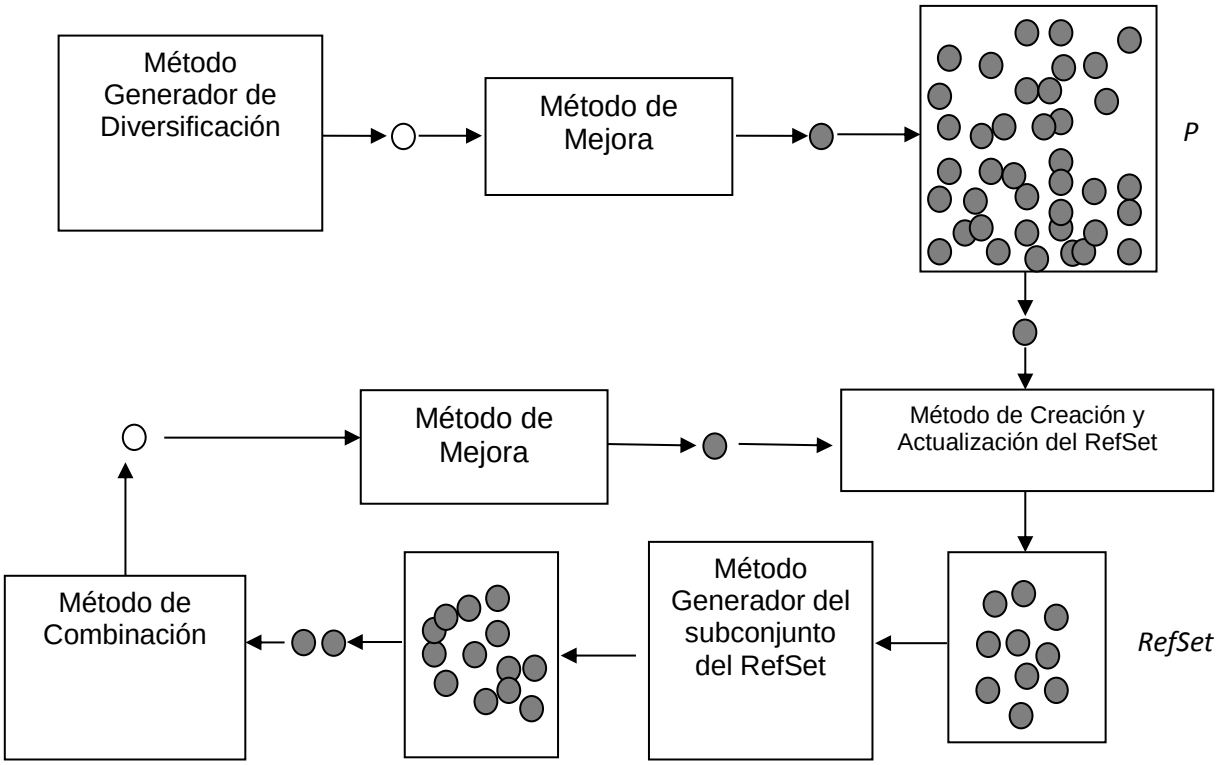

Figura 5.1 Algoritmo Scatter Search

Es importante no olvidar que el problema que nosotros intentamos resolver con este algoritmo es un problema de programación binivel, lo que implica que se debe de considerar tanto el problema del líder como el problema del seguidor debido a que hay que tomar en cuenta la decisión óptima del seguidor, es decir, que una solución factible binivel es aquella que considera la solución óptima del nivel inferior. En los métodos que conforman un algoritmo basado en Scatter Search no se puede pasar por alto éste vínculo, ya que es necesario conocer la reacción del seguidor una vez que el líder toma una decisión.

A continuación se describen detalladamente cada uno de los métodos del algoritmo; se debe de tomar en cuenta que los métodos varían de acuerdo al problema que se desea resolver, es decir, cada método se debe adecuar al problema no olvidando la meta o el objetivo de cada uno.

5.1.1 Método generador de diversificación

El objetivo de éste método es generar una población de buenas soluciones iniciales; proponemos la implementación del procedimiento de la búsqueda voraz aleatoria adaptativa (GRASP por sus siglas en inglés) como método de construcción. El GRASP básicamente consta de dos fases en cada iteración; la primera es la fase constructiva en la que se obtiene una buena solución inicial y la fase de búsqueda local donde se mejora esa solución siempre y cuando sea posible. Una descripción detallada del GRASP y varias de sus aplicaciones se pueden encontrar en Resende y González Velarde [24].

La finalidad del método constructivo es crear una solución factible inicial, la cual se va construyendo de forma iterativa agregando un elemento que cumple con ciertas características en cada iteración hasta que la solución alcance la cardinalidad deseada, es decir, hasta que se abran un número p de plantas. Para decidir qué elementos formarán parte de la solución se recurre a una función voraz que mide la contribución local de cada elemento a la solución parcial. Para el problema analizado en esta tesis, se considera dicha función como la suma de los costos de instalación y de distribución de la localización de las plantas, esto con el fin de que se refleje cuáles plantas tienen una contribución más favorable a la función objetivo. Se debe recordar que para calcular el costo miope de cada planta candidata se debe resolver el problema del nivel inferior.

Se crea una lista de candidatos en base a dicha contribución. Debido que contamos con un nivel inferior que debe resolverse de forma exacta sólo se trabaja con un porcentaje del número total de plantas candidatas posibles, éstas plantas candidatas se seleccionan uniformemente al azar.

Las mejores plantas candidatas se identifican y se crea una lista restringida de candidatos con el fin de reducir el número de candidatos potenciales que pueden ser agregados a la solución. El método de construcción se basa en el valor de la función objetivo del líder por lo que es necesario conocer los valores máximos y mínimos de

las evaluaciones, es decir, C_{min} y C^{max}. Calculamos el valor de $C_{min} + \alpha(C^{max} - C_{min})$ para cada planta candidata, si el costo de la planta candidatas es menor o igual que ese valor calculado entonces esa planta pasa a la lista restringida de candidatos. El parámetro α determina el nivel de aceptación de las plantas candidatas, donde $0 \leq \alpha \leq 1$. Se puede ver fácilmente que si $\alpha = 0$ el método tendría un comportamiento voraz y si $\alpha = 1$ el comportamiento sería completamente aleatorio.

La selección de los elementos de la lista restringida de candidatos se lleva a cabo uniformemente al azar por lo que no necesariamente se selecciona al mejor, éste es el componente probabilista del GRASP. El procedimiento es adaptativo puesto que los costos asociados a los elementos se actualizan en cada iteración con el fin de reflejar los cambios ocasionados por la selección del elemento anterior. Todo el proceso se repite hasta un criterio de paro, en este caso, hasta que la solución tenga la cardinalidad p deseada.

Las soluciones generadas en el método constructivo del GRASP son buenas soluciones factibles pero siempre en benéfico recurrir a una búsqueda local. Debido a la naturaleza de nuestro problema no es oportuno insertar una planta nueva o eliminar una que ya forma parte de la solución ya que no sería factible puesto que hay que abrir un número p de plantas, por tal motivo sólo se recurre a la estrategia de intercambiar plantas, esto es, intercambiar una planta que está en la solución por otra que no lo está.

Se crea una lista de candidatos de las plantas que no están en la solución para analizar el movimiento de intercambio, nuevamente se trabaja únicamente con un porcentaje debido al costo computacional de resolver el nivel inferior. La mitad de ese porcentaje son candidatos de calidad que se seleccionan en base a su contribución a la función objetivo y la otra mitad son elementos seleccionados al azar para garantizar la diversidad en el conjunto *Población*.

Para calcular la contribución a la función objetivo y seleccionar los candidatos de mejor calidad se recurre a una función diseñada de tal forma que para cada cliente $j \in J$ se toma el costo de distribución máximo, llamado c_{max_j}, después para cada

planta $i \in I$ y para cada cliente $j \in J$ se calcula el cociente $\frac{c_{max_j} - c_{ij}}{p_{ij}}$. Luego se examina para cada planta $i \in I$ dichos cocientes y se toma la razón más grande puesto que la diferencia $c_{max_j} - c_{ij}$ más pequeña dará un costo alto al ser c_{ij} un valor muy cercano al c_{max_j}, más aún, cuando ambos costos coinciden es claro ver que la diferencia es cero. También hay que recordar que las preferencias están ordenadas donde las más preferidas son los valores más pequeños, entonces lo que buscamos es un cociente entre un número grande (más barato) y un número pequeño (más preferido) por lo que buscamos que ese cociente o razón sea un número grande. Se ordenan de forma decreciente dichas razones y se selecciona el número de candidatos que se requieran.

Una vez seleccionada la planta candidata se analiza el intercambio, este intercambio se realiza sólo si el costo de ese intercambio es mejor que el de la solución original, si es así se procede con este movimiento de mejora y se actualiza la solución. La búsqueda local termina cuando ya no existe mejora posible. A continuación se presenta en la Figura 5.2 el pseudocódigo básico del GRASP que utilizamos, en el Anexo 1 se encuentra el pseudocódigo de la adaptación diseñada del GRASP genérico de manera más detallada.

Procedimiento GRASP

$y_k \leftarrow \emptyset$
$Lista \leftarrow \emptyset$
$stop \leftarrow \emptyset$
while $stop \leq PSize$
    $y_k \leftarrow Construcción\ voraz\ aleatoria\ adaptativa\ (\ \ )$
    $y'_k \leftarrow Búsqueda\ local\ (y_k)$

    if $c(y'_k) < c(y_k)$
        $Lista \leftarrow Lista \cup \{y'_k\}$

Figura 5.2 Pseudocódigo general de GRASP

```
            else
                Lista ← Lista ∪ {y_k}
            end if

        Actualizar PSize (Lista)

end while
return Población

end GRASP
```

Figura 5.2 Pseudocódigo general de GRASP (continuación)

5.1.2 Método de creación y actualización del conjunto referencia

Una vez creado el conjunto de soluciones iniciales se crea un subconjunto tomando b soluciones de $Población$ que pasarán a formar el $RefSet$, dichas soluciones deben ser diversas y de calidad. Para cumplir este criterio, se toman las $b/2$ mejores soluciones de la $Población$ en base a la evaluación de la función objetivo y las otras $b/2$ se extraen de $Población$ maximizando de forma dinámica la mínima distancia de las soluciones ya incluidas en el $RefSet$, es decir, se calcula la mínima distancia que hay entre las $b/2$ mejores soluciones de $Población$ que fueron introducidas en $RefSet$ y el resto de soluciones que pertenecen a la $Población$, después se introduce al $RefSet$ la solución que tenga la distancia más grande dentro de las mínimas distancias que se calcularon; éste proceso se repite pero ahora considerando en el cálculo de las distancias las soluciones que van perteneciendo al $RefSet$. Debido a la representación de la solución del líder, la métrica de la distancia se considera como la suma de las diferencias entre las variables de la solución, es decir:

$$d(y^P, y^{RS}) = \sum_{i \in I} \left| (y_i^P - y_i^{RS}) \right| \tag{5.1}$$

donde y^P es la solución que se encuentra en el conjunto $Población$ y y^{RS} es la solución que pertenece al $RefSet$. Esta métrica se calcula para todas las soluciones que se encuentran en el $RefSet$.

Es así como se crea el $RefSet$, pero es necesario que éste conjunto se actualice, para esto, cada solución combinada entrará al conjunto sólo si mejora a la peor solución que ya está dentro. Debido a que se introduce una solución sólo si sale otra, el $RefSet$ se mantiene de tamaño constante. El método se detiene, es decir, el conjunto $RefSet$ deja de actualizarse cuando al tratar de combinar soluciones no existen elementos nuevos en éste conjunto. Pero antes que las soluciones de éste conjunto pasen por el método de combinación es necesario que pasen por otro método donde se agruparán las soluciones para que puedan ser combinadas. A continuación se describen ambos métodos.

5.1.3 Método generador del subconjunto del conjunto referencia

Teniendo ya creado el conjunto referencia, se crea otro subconjunto donde estarán todas las parejas de soluciones combinadas del $RefSet$ que pasarán por el método de combinación. Debido a que éste método se aplicará más de una vez, es importante aclarar que sólo entrarán a este conjunto las soluciones que no han sido combinadas anteriormente.

5.1.4 Método de combinación

Una vez creado el subconjunto del $RefSet$ con todas las parejas de soluciones posibles se procede a aplicar un método de combinación con el fin de crear nuevas soluciones. El método de combinación que consideramos utiliza una regla de probabilidad diseñada con la finalidad de que la solución resultante de la combinación

sea mejor que las que ya pertenecen al conjunto referencia; consideramos una puntuación asociada a cada componente del par de soluciones combinadas; la puntuación se puede interpretar como la probabilidad de que la variable tome el valor de uno. Esto es, para cada elemento de las soluciones combinadas se calcula una puntuación llamada $score_i$:

$$score_i = \frac{y_i^w VFO^w + y_i^z VFO^z}{VFO^w + VFO^z} \tag{5.2}$$

donde y_i^w es el valor de la $i-ésima$ variable de la solución w y VFO^w es el valor de la función objetivo de la solución y_i^w, y la y_i^zes el valor de la $i-ésima$ variable de la solución z y su valor de la función objetivo es la VFO^z.

Se genera un vector $rand_i$ de números aleatorios, donde $0 \leq rand_i \leq 1$. Si $rand_i \leq score_i$ entonces el componente correspondiente de la nueva solución combinada tomará el valor de 1, es decir, $y_i = 1$; en caso contrario tomará el valor de cero esa posición, esto es, $y_i = 0$. Un ejemplo de este método de combinación se puede ver en la Figura 5.3.

$y^1 = [0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0]$ $\quad VFO^1 = 1320239.75$

$y^2 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0]$ $\quad VFO^2 = 1329914.25$

$score = [0.502\ 0.000\ 0.000\ 0.499\ 0.000\ 0.499\ 0.000\ 1.000\ 0.502\ 0.000]$

$rand = [0.308\ 0.152\ 0.520\ 0.179\ 0.780\ 0.252\ 0.260\ 0.180\ 0.329\ 0.845]$

$y_{combinada} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0]$

Figura 5.3 Método de Combinación

Se puede dar el caso de que se generen soluciones infactibles combinando dos soluciones factibles; en nuestro problema, la infactibilidad puede surgir al combinar dos soluciones que originen una nueva solución que no cuente con la cardinalidad deseada. Esto no resulta inconveniente porque las soluciones que resultan de las combinaciones pasan por un método de mejora donde se les devuelve la factibilidad, dicho método es el mismo al que recurrimos en el GRASP. El método se detiene

cuando al tratar de combinar soluciones no existen elementos nuevos en el conjunto referencia.

5.1.5 Método de mejora

El objetivo de éste método es mejorar la calidad de las soluciones combinadas y se divide en dos fases: factibilidad y mejora.

En la fase de factibilidad se asegura que la solución combinada tenga exactamente cardinalidad p. Una solución puede ser infactible después de pasar por el método de combinación porque puede tener más o menos plantas localizadas p, esto se puede apreciar en la solución combinada mostrada en la Figura 5.3. Con el fin de obtener una solución factible, en esta fase se procede con dos movimientos: abrir o cerrar las plantas necesarias hasta que se alcance el número p deseado. El criterio que se consideró para abrir plantas es el mismo que se utilizó en la fase inicial de construcción. Por otro lado, para cerrar plantas, se calcula el impacto que la función objetivo del líder tendrá después de eliminar una planta. La planta que da una mejor reducción en el valor de la función objetivo se eliminará y se actualizará la solución.

En caso de que la solución sea factible, entrará directo a la fase de mejora que es el mismo procedimiento descrito en la búsqueda local aplicada en el GRASP donde sólo se recurre al movimiento de intercambio de plantas. Un ejemplo de éste método de mejora se puede apreciar en la Figura 5.4.

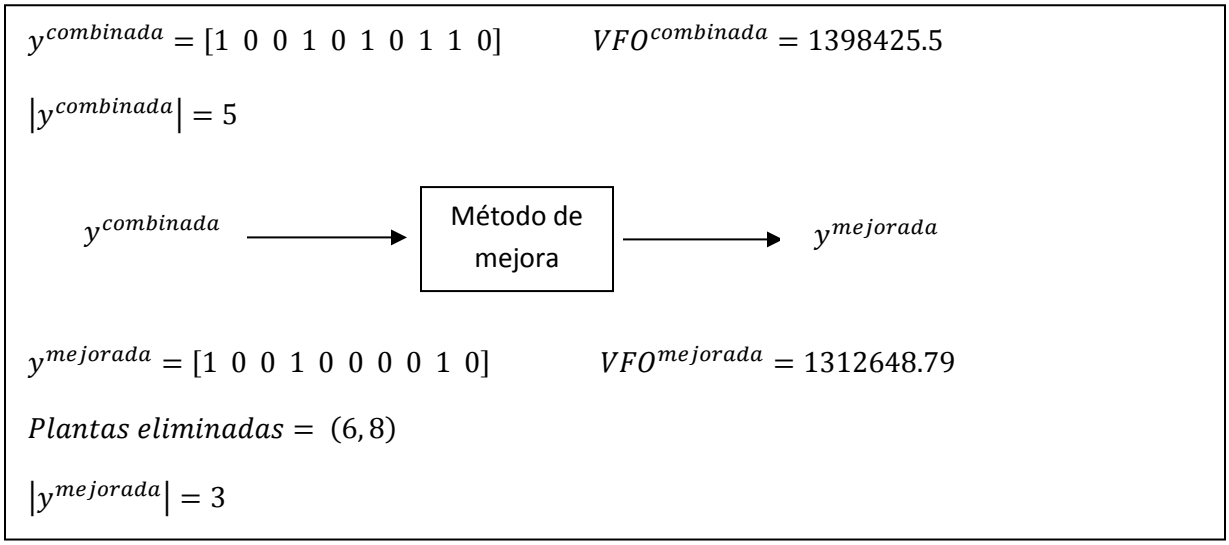

Figura 5.4 Método de Mejora

Con el fin de evitar la duplicación de las soluciones de la $Población$ y del $RefSet$, se codifica cada solución con la siguiente función hash:

$$H(y) = \sum_{i \in I} y_i 2^i \tag{5.3}$$

El valor de hash y el de la función objetivo de cada solución se almacenan. De esta manera sólo se compara el valor hash de la nueva solución con los valores de las soluciones que ya pertenecen a la $Población$ o al $RefSet$, si es igual indicará que la solución ya está dentro y se descarta, en caso contrario, la solución podrá ser introducida en alguno de los dos conjuntos mencionados anteriormente. Se actualizan tanto el valor de hash como el de la función objetivo de las soluciones que están dentro de la $Población$ o del $RefSet$. Se aplica este filtro de evaluación de la función hash en todo el algoritmo, es decir, en el método generador de diversificación, en el método de creación y actualización del conjunto referencia y en el método de combinación. Este procedimiento garantiza que todas las soluciones tanto de la $Población$ como del $RefSet$ serán siempre diferentes y no se trabajará al doble al volver a evaluar una solución que ya se había calculado su función objetivo.

CAPÍTULO 6

EXPERIMENTACIÓN COMPUTACIONAL

En este capítulo se describen las instancias que utilizamos para validar las reformulaciones y las variantes propuestas del algoritmo así como el ambiente computacional. Al final del capítulo se muestran los resultados de la experimentación numérica.

6.1 INSTANCIAS DE PRUEBA

Se realizó un análisis con el fin de evaluar el desempeño de las dos reformulaciones mostradas en el capítulo 4 y del algoritmo propuesto mostrado en el capítulo 5. Para esta evaluación computacional consideramos el conjunto de instancias reportados en Cánovas et al. [10]. Dentro de este conjunto de instancias, se probaron tres subconjuntos diferentes de 12 problemas diferentes. Dichos subconjuntos se clasificaron de acuerdo al tamaño de las instancias: pequeños, medianos y grandes. Las instancias de tamaño pequeño consisten en 50 plantas y 50 clientes (50x50), las instancias medianas y grandes tienen las dimensiones de 50x75 y 75x100, respectivamente.

En Cánovas et al. [10] resuelven los casos para el problema de una sola planta con preferencias ordenadas y muestran el valor óptimo de la función objetivo así como el número óptimo de plantas por abrir, esto nos sirve como punto de referencia para hacer las comparaciones y para analizar el problema de la p-Mediana. Es

importante notar que el óptimo de esas instancias es conocido pero no hay información sobre la forma o tiempo que se requirió para obtenerlo.

6.2 AMBIENTE COMPUTACIONAL

Los experimentos se llevaron a cabo en una computadora personal modelo HP Compaq 6000 Pro SFF PC con procesador Pentium(R) Dual-Core, de 3.0GHz, con memoria RAM de 2.00 GB y Windows 7 Professional. Los códigos fueron implementados en Visual Studio 2010 con lenguaje C++.

Ambas reformulaciones presentadas en el capítulo 4 se resolvieron de manera directa con CPLEX 12.1 en lenguaje de programación C++.

6.3 RESULTADOS COMPUTACIONALES

A continuación se presentan primero los resultados de las dos reformulaciones y después los resultados del algoritmo propuesto.

6.3.1 REFORMULACIONES

En las tablas 6.1, 6.2 y 6.3 se presentan los resultados de las dos reformulaciones propuestas. La primera columna, "Instancia", representa el nombre o etiqueta de cada instancia. La segunda columna, "p", indica el número de plantas por abrir en cada instancia. La tercera columna, "Óptimo", indica el valor óptimo de la función objetivo del nivel superior. Las dos columnas restantes muestran el tiempo requerido por ambas reformulaciones para resolver de manera óptima las instancias, donde "Ref.1" representa la reformulación basada en las relaciones primal-dual del problema del nivel inferior y "Ref.2" representa la reformulación donde se sustituye el

nivel inferior con un conjunto de restricciones para que las preferencias de los clientes se sigan tomando en cuenta en el problema.

Instancia	p	Óptimo	Ref.1	Ref.2
132_1	8	1,122,750	20,556.07	15.114
132_2	9	1,157,722	108,000.00	16.493
132_3	6	1,146,301	20,939.78	24.959
132_4	5	1,036,779	3,805.14	9.162
133_1	7	1,103,272	11,507.38	11.585
133_2	5	1,035,443	828.33	9.362
133_3	6	1,171,331	5,865.24	14.184
133_4	9	1,083,636	22,737.36	14.189
134_1	4	1,179,639	5,035.57	14.991
134_2	7	1,121,633	3,719.37	13.340
134_3	6	1,171,409	10,664.96	31.397
134_4	3	1,210,465	2,296.86	11.876

Tabla 6.1 Tiempo requerido (en segundos) para las instancias de 50x50

En los casos donde la marca "*" aparece, indica que la computadora se quedó sin memoria y el optimizador se detiene con el mejor valor encontrado hasta este momento. Es importante mencionar que se estableció para ambas reformulaciones como tiempo máximo 30 horas para la resolución de los problemas. Las instancias donde CPLEX no fue capaz de resolver a optimalidad y consumió todo el tiempo permitido se muestran en las tablas teniendo 108,000 segundos en la posición correspondiente.

Instancia	p	Óptimo	Ref.1	Ref.2
a75_50_1	7	1,661,269	108,000.00	675.402
a75_50_2	6	1,632,907	108,000.00	386.389
a75_50_3	7	1,632,213	108,000.00	430.570
a75_50_4	5	1,585,028	108,000.00	451.094
b75_50_1	8	1,252,804	108,000.00	338.354
b75_50_2	9	1,337,446	108,000.00	418.605
b75_50_3	9	1,249,750	108,000.00	460.500
b75_50_4	9	1,217,508	108,000.00	269.313
c75_50_1	11	1,310,193	*	350.672
c75_50_2	10	1,244,255	108,000.00	266.737
c75_50_3	12	1,201,706	*	238.057
c75_50_4	11	1,334,782	108,000.00	311.506

Tabla 6.2 Tiempo requerido (en segundos) para las instancias de 50x75

De las tablas 6.1-6.2 se puede observar que la primera reformulación consume más tiempo que la segunda reformulación. La principal razón de esta diferencia significativa en el tiempo computacional es que la reformulación en base a las condiciones de optimalidad primal-dual implica mucho más variables y restricciones, mientras que la otra reformulación propuesta incorpora el problema del nivel inferior de una manera diferente y equivalente por lo cual se puede reescribir el problema de forma más compacta que en la reformulación del problema binivel.

Para la instancia 132_2 se llegó a una holgura de 2.2% cuando el optimizador se detuvo debido que se alcanzó el tiempo máximo permitido. Para los casos 50x75, CPLEX alcanza el tiempo máximo permitido en 10 de 12 casos donde siempre tuvo una holgura mayor que 30%. Para las instancias c75_50_1 y c75_50_3 la computadora se quedó sin memoria después de aproximadamente 19 y 24 horas respectivamente.

Instancia	p	Óptimo	Ref.1	Ref.2
a100_75_1	4	2,286,397	*	5,160.935
a100_75_2	3	2,463,187	*	3,371.245
a100_75_3	3	2,415,836	*	4,096.929
a100_75_4	4	2,380,150	*	11,232.289
b100_75_1	8	1,950,231	*	46,388.404
b100_75_2	8	2,023,097	*	93,604.671
b100_75_3	8	2,062,595	*	58,456.122
b100_75_4	9	1,865,323	*	45,577.674
c100_75_1	6	1,843,620	*	13,547.217
c100_75_2	11	1,808,867	*	18,535.715
c100_75_3	8	1,820,587	*	59,725.313
c100_75_4	9	1,839,007	*	25,689.271

Tabla 6.3 Tiempo requerido (en segundos) para las instancias de 75x100

En la tabla 6.3 se puede observar que la computadora se quedó sin memoria al ejecutar la reformulación basada en las relaciones primal-dual del problema del nivel inferior y que el tiempo mínimo registrado en la segunda reformulación propuesta es el de la instancia a100_75_2 que fue casi de una hora pero en cambio, en la instancia b_100_75_2 se registró un tiempo de 26 horas lo cual es un tiempo considerable. Los resultados muestran la necesidad de diseñar e implementar otras metodologías para

la resolución de este problema, por tal motivo proponemos un algoritmo heurístico que busca obtener resultados de buena calidad en un tiempo razonable; dichos resultados se presentan a continuación.

6.3.2 ALGORITMO STACKELBERG-SCATTER SEARCH

Una de las ventajas del algoritmo implementado es que sólo utiliza dos parámetros: el grado de aleatoriedad en la lista restringida de candidatos y el número de plantas candidatas que se evalúan con el fin de ingresar a la lista de candidatos. Después de pruebas preliminares, los resultados mostraron que un $\alpha = 0.4$ proporciona buenas soluciones en cuestión de calidad y de diversidad y, 6 candidatos parecía ser un valor eficiente para los tres tamaños de todas las instancias. Debido a que el algoritmo propuesto tiene aleatoriedad, cuando se resolvía una instancia podríamos obtener dos valores diferentes en dos diferentes corridas, es por esto, que se decidió realizar 10 corridas para cada instancia con los parámetros seleccionados.

En las tabla 6.4, 6.5 y 6.6 se presentan los resultados. La primera columna, "Instancia", representa el nombre o etiqueta de cada instancia. La segunda columna, "p", indica el número de plantas a abrir en cada instancia. La tercera columna, "Óptimo", indica el valor óptimo de la función objetivo del nivel superior. En la columna "Ref.2" se muestra el tiempo requerido para resolver la reformulación del p-Mediana BPO de manera óptima, éste tiempo es el que se mostró en la sección anterior. Finalmente, en los últimas cuatro columnas se describen los resultados numéricos obtenidos después de correr 10 veces el algoritmo Stackelberg-Scatter Search (denotado en las tablas por "S-SS"). Estas columnas representan el valor promedio de la función objetivo del líder, la desviación estándar, el tiempo requerido (en segundos) y el porcentaje de veces que el algoritmo encontró la solución óptima de la instancia. Aquí sólo reportamos el promedio del valor de la función objetivo pero en el Anexo 2 se pueden encontrar los resultados de las 10 corridas de todas las instancias.

			Ref.2	S-SS			
Instancia	***p***	**Óptimo**	**Tiempo**	**Promedio**	**Desv. Est.**	**Tiempo**	**%**
132_1	8	1,122,750	15.114	1,135,633.588	16,364.181	2.540	50
132_2	9	1,157,722	16.493	1,186,726.238	16,254.201	2.170	20
132_3	6	1,146,301	24.959	1,170,880.875	26,248.841	2.016	50
132_4	5	1,036,779	9.162	1,054,174.357	28,604.396	2.100	70
133_1	7	1,103,272	11.585	1,119,174.450	28,818.973	2.275	60
133_2	5	1,035,443	9.362	1,071,267.538	44,128.754	4.035	50
133_3	6	1,171,331	14.184	1,203,326.450	24,979.020	1.994	30
133_4	9	1,083,636	14.189	1,088,738.525	4,703.919	2.037	30
134_1	4	1,179,639	14.991	1,195,964.400	26,894.654	2.216	70
134_2	7	1,121,633	13.340	1,146,708.875	19,675.166	2.107	30
134_3	6	1,171,409	31.397	1,184,028.688	12,113.238	1.916	40
134_4	3	1,210,465	11.876	1,221,040.675	13,973.978	2.774	50

Tabla 6.4 Resultados de las instancias de tamaño 50x50

En la tabla 6.4 se puede observar que el tiempo requerido para la resolver las instancias de tamaño pequeño con la reformulación es aceptable. Además, los resultados muestran que el algoritmo binivel propuesto tiene un buen rendimiento, ya que en más de 45% de las veces que se corrió alcanzó el valor óptimo para los 12 casos. La desviación estándar indica que todas las soluciones dadas por el algoritmo son aceptables respecto al mejor valor objetivo encontrado. Por otra parte, los tiempos requeridos se reducen aproximadamente un 82% con respecto al tiempo necesario para la resolución de la reformulación de un solo nivel.

			Ref.2	S-SS			
Instancia	***p***	**Óptimo**	**Tiempo**	**Promedio**	**Desv. Est.**	**Tiempo**	**%**
a75_50_1	7	1,661,269	675.402	1,685,001.913	26,352.410	3.629	50
a75_50_2	6	1,632,907	386.389	1,646,106.213	24,233.844	3.554	70
a75_50_3	7	1,632,213	430.570	1,640,945.750	12,579.997	3.791	40
a75_50_4	5	1,585,028	451.094	1,598,163.038	27,900.146	2.603	80
b75_50_1	8	1,252,804	338.354	1,283,771.000	41,413.418	3.700	60
b75_50_2	9	1,337,446	418.605	1,342,658.225	5,395.184	3.765	40
b75_50_3	9	1,249,750	460.500	1,277,441.013	17,178.253	4.608	20
b75_50_4	9	1,217,508	269.313	1,224,684.175	9,361.354	3.081	60
c75_50_1	11	1,310,193	350.672	1,324,409.950	11,793.501	4.473	30
c75_50_2	10	1,244,255	266.737	1,257,343.425	12,217.801	3.708	20
c75_50_3	12	1,201,706	238.057	1,227,030.013	30,517.698	4.168	40
c75_50_4	11	1,334,782	311.506	1,340,215.113	7,635.247	4.041	60

Tabla 6.5 Resultados de las instancias de tamaño 50x75

En la tabla 6.5 se presentan los resultados de la experimentación realizada con las instancias medianas. La reformulación de un solo nivel aumenta en tiempo computacional debido a que tarda más de 6 minutos en promedio en resolver cada instancia. Por otro lado, la metaheurística desarrollada mantiene un tiempo aceptable, ya que alcanza el valor óptimo en casi la mitad de las 120 corridas.

			Ref.2	S-SS			
Instancia	***p***	**Óptimo**	**Tiempo**	**Promedio**	**Desv. Est.**	**Tiempo**	**%**
a100_75_1	4	2,286,397	5,160.935	2,349,382.417	76,503.312	11.280	50
a100_75_2	3	2,463,187	3,371.245	2,477,142.175	23,045.960	10.357	70
a100_75_3	3	2,415,836	4,096.929	2,430,237.375	16,016.482	7.566	40
a100_75_4	4	2,380,150	11,232.289	2,403,067.850	35,534.875	7.937	60
b100_75_1	8	1,950,231	46,388.404	1,982,068.500	19,494.323	13.499	20
b100_75_2	8	2,023,097	93,604.671	2,056,043.388	14,214.883	15.153	10
b100_75_3	8	2,062,595	58,456.122	2,079,068.800	14,336.538	13.846	30
b100_75_4	9	1,865,323	45,577.674	1,909,573.925	26,532.900	11.004	20
c100_75_1	6	1,843,620	13,547.217	1,852,251.163	13,929.673	11.141	70
c100_75_2	11	1,808,867	18,535.715	1,831,758.125	18,124.272	12.573	30
c100_75_3	8	1,820,587	59,725.313	1,831,926.275	13,117.874	10.503	50
c100_75_4	9	1,839,007	25,689.271	1,858,127.350	18,951.886	12.244	30

Tabla 6.6 Resultados de las instancias de tamaño 75x100

De la misma manera que en las dos últimas tablas, la tabla 6.6 muestra los resultados de la experimentación computacional de las instancias de tamaño grande. Se puede observar que es muy ineficiente resolver los casos con la reformulación de un solo nivel. El tiempo mínimo reportado fue de más de 56 minutos para encontrar el valor óptimo en la instancia a100_75_2. Por otra parte, la metaheurística Scatter Search también aumenta su tiempo computacional pero no tan drásticamente como la reformulación de un solo nivel. El tiempo reportado varía de 7.5 a 15.2 segundos con un promedio de 11.43 segundos. Por lo tanto, el algoritmo propuesto alcanza el óptimo en al menos una de las 10 corridas en las 12 instancias. La columna de la desviación estándar muestra que el algoritmo tiene un rendimiento estable que siempre alcanza valores cercanos al promedio de las 10 corridas para cada instancia. El único valor atípico corresponde a la instancia a100_75_1 pero el valor óptimo se alcanza en 5 de las 10 corridas, a pesar de la clara existencia de la aleatoriedad en la

columna de la desviación estándar se pudo llegar al óptimo en el 50% de las corridas lo que indica que el algoritmo no tiene un mal desempeño.

En la Figura 6.7 se muestra la gráfica del tiempo requerido para resolver cada una de las 36 instancias, es el tiempo reportado en las columnas "Ref.2" y "S-SS" de las tablas 6.4-6.6. El conjunto de instancias lo dividimos en tres subconjuntos diferentes respecto al tamaño de las instancias, los resultados aparecen representados de manera agrupada. Los primeros 12 datos corresponden a las instancias de tamaño chico, los datos del 13 al 24 representan los resultados de las instancias de tamaño mediano, y los 12 datos finales están asociados con las instancias de tamaño grande. Para poder visualizar correctamente los resultados, los datos se representan en una escala logarítmica. Debido a la magnitud de los tiempos reportados con la reformulación Ref.2 de las instancias de 75x100 la escala del eje se amplía. Los círculos rojos indican el tiempo promedio requerido para cada grupo asociado con un tamaño particular de instancia. En la gráfica se puede observar que a pesar del hecho de que el algoritmo basado en la metaheurística Scatter Search aumenta su tiempo requerido para resolver las instancias, éste es un aumento polinomial; mientras que el incremento correspondiente a la reformulación del problema a un solo nivel parece ser exponencial.

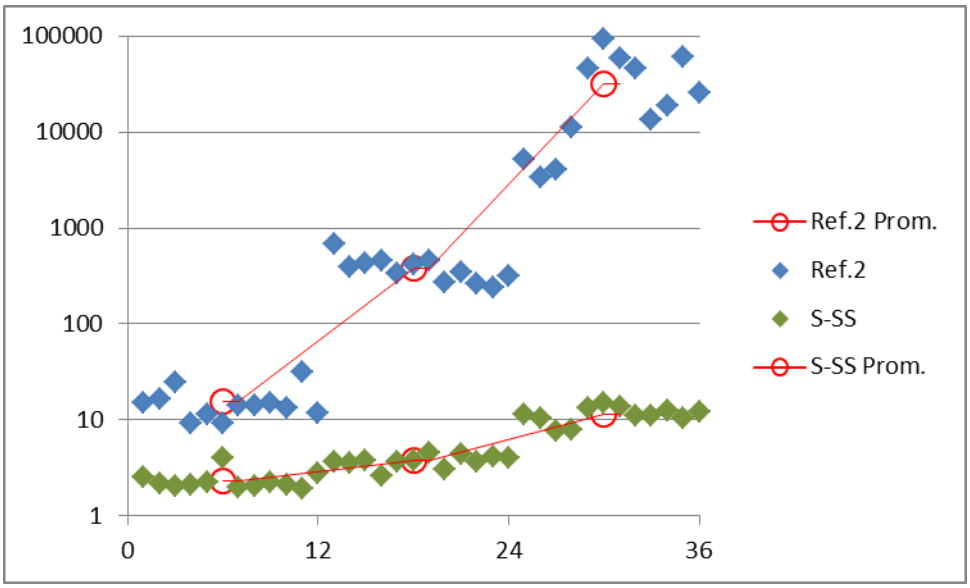

Figura 6.7 Tiempo computacional requerido para resolver las instancias

CAPÍTULO 7

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se presentan las conclusiones a las que llegamos después de estudiar detalladamente el problema binivel de la p-Mediana con orden y de analizar los resultados arrojados en la experimentación computacional de las reformulaciones así como del algoritmo que proponemos. Además, se presentan las posibles líneas de investigación futura que se pueden desarrollar a partir del trabajo presentado en esta tesis.

7.1 CONCLUSIONES

En esta tesis consideramos el problema binivel de la p-Mediana con orden donde examinamos el caso en el cual las plantas no tenían capacidades. Debido a que se formuló el problema con programaciónn binivel consideramos que el objetivo del líder es minimizar los costos de instalación y de distribución; y el seguidor optimiza las preferencias ordenadas de los clientes. El objetivo del problema es localizar las p plantas que minimicen el costo total (instalación y distribución) de localización y, a su vez, minimicen las preferencias ordenadas de los clientes.

Con el fin de resolver el problema, se proponen dos nuevas formulaciones de un solo nivel y un algoritmo para el problema binivel. La primera reformulación está basada en las relaciones primal-dual del problema del seguidor y la segunda reformulación se basa en agregar un conjunto de restricciones para que se sigan

considerando las preferencias de los clientes. Los resultados numéricos demuestran que el uso de un software de optimización para resolver la primera reformulación es ineficiente (en términos de los tiempos requeridos para resolver las instancias) para todas las instancias ya que algunas se consume toda la memoria de la computadora. Por otro lado, la segunda reformulación requiere menos tiempo que la primera pero cuando resuelve las instancias de tamaño mediano y grande consume una cantidad significativa de tiempo, 5.942 minutos y 535.258 minutos en promedio respectivamente.

Como resultado del tiempo requerido en la experimentación computacional de ambas reformulaciones proponemos un algoritmo heurístico para resolver el problema, recurrimos a otra alternativa para resolver el nivel inferior en lugar de resolverlo de forma exacta con CPLEX. El algoritmo está basado en la metaheurística de Scatter Search y considera el equilibrio de Stackelberg durante el proceso, es decir, se obtiene la respuesta óptima del seguidor para cada decisión.. El objetivo de este algoritmo es resolver la formulación binivel del p-Mediana BPO. Este algoritmo se valida usando el mismo conjunto de instancias que el utilizado para las formulaciones propuestas. Los resultados obtenidos indican que el rendimiento de la metaheurística es muy bueno en términos del valor de la función objetivo del líder y del tiempo computacional pequeño requerido para llegar a la solución óptima de todas las instancias.

Se puede observar también en los resultados reportados que el rendimiento de la metaheurística no depende del tamaño de la instancia; desafortunadamente no se puede probar su desempeño en problemas de mayor tamaño debido a la falta de soluciones óptimas conocidas para casos más grandes del p-Mediana BPO. Más aún, suponiendo el caso donde algunas instancias existentes de un tamaño más grande que 75x100 se adaptaran mediante la adición de las preferencias ordenadas de los clientes el valor óptimo para el problema binivel podría ser desconocido y las reformulaciones propuestas no serían capaces de encontrar el óptimo debido al tiempo computacional que requerirían para resolver las instancias a optimalidad. Es importante mencionar que esto no es una debilidad del algoritmo, simplemente no tenemos como comparar

o medir la eficiencia de nuestro algoritmo debido a que actualmente no existen cotas válidas con buena holgura de optimalidad aceptables.

7.2 TRABAJO FUTURO

Una extensión natural del problema estudiado en esta tesis es considerar capacidades en las plantas y las demandas de los clientes. Hasta donde sabemos, no ha sido abordado en la literatura, consideramos que esto se debe, en gran parte, a que bajo el enfoque binivel el nivel inferior resultante (por sí mismo) es un problema NP-Hard. Al incorporar esas restricciones el problema del nivel inferior se convertiría en un Problema de Asignación Generalizada (GAP, por sus siglas en inglés), el cual es un problema muy conocido y estudiado debido a su complejidad y propiedades. Se ha demostrado que este problema es NP-Hard, por lo tanto la solución óptima del problema del nivel inferior no se puede calcular de manera sencilla. La complejidad del problema binivel considerando capacidades y demandas se basa en el hecho de que en el esquema de resolución clásica se necesita obtener la mejor respuesta del seguidor para cada decisión del líder. Esto podría no ser posible porque el nivel inferior es un problema NP-Hard, por lo tanto usar una heurística sería muy costoso en términos de tiempo computacional debido al gran número de corridas que la heurística podría llevar a cabo para resolver el problema del seguidor. Se pueden explotar las propiedades del GAP para tratar con el nivel inferior. En este caso, se tienen que introducir los conceptos necesarios para tener una región inducida aproximada que sea eficiente. Además, se pueden desarrollar e implementar algoritmos mateheurísticos para el problema completo, que consideren las propiedades estudiadas del GAP dentro de su metodología.

Otra posible extensión del trabajo presentado es considerar la eliminación de la restricción que indica el número de plantas por abrir, es decir, la restricción que convierte el problema en uno de la p-Mediana. El principal problema que podemos ver es que en la construcción de las soluciones por medio del GRASP se tendrían

complicaciones al momento de detener la apertura de plantas puesto que no se conoce el número a abrir. Por otro lado, habíamos pensado que un algoritmo basado en adaptive memory para crear las soluciones sería muy eficiente. Este algoritmo podría compararse contra el algoritmo evolutivo existente en la literatura. De esta manera, también se podría considerar tanto el caso sin capacidades como el caso donde los clientes demandas y las plantas tengan capacidades.

BIBLIOGRAFÍA

[1] A. Weber, Theory of the Location of Industries, *University of Chicago Press*, Chicago, Ill, USA, Translated by C.J. Friedrich, 1929.

[2] Z. Drezner, H. Hamacher, Facility Location: Applications and Theory, Springer-Verlag, Berlin, 2002.

[3] G. Sahin, H. Süral, A Review of Hierarchical Facility Location Models, *Computers & Operations Research*, vol. 34, pp. 2310 – 2331, 2007.

[4] M.T. Melo, S. Nickel, F. Saldanha-da-Gama, Facility location and supply chain management – A review, *European Journal of Operational Research*, vol. 196, pp. 401-412, 2009.

[5] A.A. Kuehn, M.J. Hamburger, A heuristic program for locating warehouses, *Management Science*, vol. 9, pp. 643-666, 1963.

[6] M.S. Daskin, Network and Discrete Location: Models, Algorithms and Applications, *Wiley-Interscienee Publication John Wiley & Sons, Inc.*, 1995.

[7] S.L. Hakimi, Optimum locations of switching centers and the absolute centers and medians of a graph, *Operations Research*, vol. 12, pp. 450–459, 1964.

[8] P. Hanjoul, D. Peeters, Facility location problem with clients' preference orderings, Regional Science and Urban Economics, vol. 17, pp. 451-473, 1987.

[9] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi , P. Crescenzi, V. Kann, Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, Springer, Berlin, 1999.

[10] L. Cánovas, S. García, M. Labbé, A. Marín, A strengthened formulation for the simple plant location problem with order, *Operations Research Letters*, vol. 35, no. 2, pp.141-150, 2007.

[11] P. Hansen, Y. Kochetov, N. Mladenovic, Lower bounds for the uncapacitated facility location problem with user preferences, *Preprint G-2004-24, Mart 2004, GERAD-HEC*, Montreal, Canada, 2004.

[12] I.L. Vasil'ev, K.B. Klimentova, Y.A. Kochetov, New lower bounds for the facility location problem with clients' preferences, *Computational Mathematics and Mathematical Physics*, vol. 49, no. 6, pp. 1010–1020, 2009.

[13] I.L. Vasil'ev, K.B. Klimentova, The branch and cut method for the facility location problem with client's preferences, *Journal of Applied and Industrial Mathematics*, vol. 4, no. 3, pp. 441–454, 2010.

[14] M. Marić, Z. Stanimirović, N. Milenković, Metaheuristic methods for solving the Bilevel Uncapacitated Facility Location Problem with Clients' Preferences, *Electronic Notes in Discrete Mathematics*, vol. 39,pp. 43-50, 2012.

[15] J.F. Camacho-Vallejo, A.E. Cordero-Franco, R.G. González-Ramírez, Solving the Bilevel Facility Location Problem under Preferences by a Stackelberg-Evolutionary algorithm, *Mathematical Problems in Engineering*. Volume 2014, Article ID 430243, 14 pages, 2014.

[16] E.V. Alekseeva, T.A. Kochetov, Genetic Local Search for eh p-Median Problem with Client's Preferences, *Diskret. Anal. Issled. Oper*.,vol. 14 no. 1, pp. 3-31, 2007.

[17] H.V. Stackelberg, Marktform und Gleichgewicht, Springer, Vienna, 1934.

[18] M. Laguna, R. Martí, Scatter Search, Kluwer Academic Publishers, Boston, 2003.

[19] S. Dempe, Foundations of Bilevel Programming, *Kluwer Academic Publishers*, 2002.

[20] J,F. Bard, Practical Bilevel Optimization: Applications and Algorithms, *Kluwer Academic Publishers*, 1998.

[21] J,F. Bard, Some Properties of the Bilevel Programming Problem, *Journal of Optimization Theory and Applications,* vol. 68, no.2, 1991.

[22] L.N. Vicente, P.H. Calamai, Bilevel and Multilevel Programming: A Bibliography Review, *Kluwer Academic Publishers*, 1994.

[23] M.S. Bazaraa, J.J. Jarvis, H.D. Sherali, Linear Programming and Network Flow, *Wiley Student Editions,* 2010.

[24] M.G.C. Resende, J.L. González Velarde, GRASP: Procedimientos de búsqueda miopes aleatorios y adaptativos, *Revista Iberoamericana de Inteligencia Artificial*, no. 19, pp. 61-76, 2003.

ANEXO 1

PSEUDOCÓDIGO DEL GRASP

Procedimiento GRASP

$y_k \leftarrow \emptyset$
$Lista \leftarrow \emptyset$
$stop \leftarrow \emptyset$
while $stop \leq PSize$
$y_k \leftarrow Construcción\ voraz\ aleatoria\ adaptativa\ (\ \ )$
$y'_k \leftarrow Búsqueda\ local\ (y_k)$

if $c(y'_k) < c(y_k)$
$Lista \leftarrow Lista \cup \{y'_k\}$
else
$Lista \leftarrow Lista \cup \{y_k\}$
end if

$Actualizar\ PSize\ (Lista)$

end while
return $Población$

end GRASP

Figura A.1.1 Pseudocódigo del GRASP

Procedimiento Construcción voraz aleatoria adaptativa

$y_k \leftarrow \emptyset$
$C \leftarrow \{ e_i \in I : e_i \, seleccionada \, uniformemente \, al \, azar\}$
Costo miope : $c(e_i) \leftarrow f_i + \sum_i \sum_j c_{i\,j} \, x_{i\,j} \quad \forall \, e_i \in C$ donde
$x_{i\,j} \in Argmin \{ \sum_i \sum_j p_{i\,j} \, x_{i\,j} : \sum_i x_{ij} = 1 \quad \forall j \in J$
$x_{ij} \leq y_k \cup \{e_i\} \quad \forall i \in I, j \in J$
$x_{ij} \in \{0,1\} \quad \forall i \in I, j \in J \}$

while $|y_k| \leq P$
$c^{max} \leftarrow \max \{c(e_i) : e_i \in C\}$
$c_{min} \leftarrow \min \{c(e_i) : e_i \in C\}$
$RCL \leftarrow \{e_i \in C : c(e_i) \leq c_{min} + \alpha \, (c^{max} - c_{min})\}$
$Seleccionar \, un \, elemento \, e_i \, de \, RCL \, uniformemente \, al \, azar$
$y_k \leftarrow y_k \cup e_i$
$Actualizar \, C$
$Calcular \, c(e_i) \, \forall \, e_i \in C$
end while
return y_k

end Construcción voraz aleatoria adaptativa

Figura A.1.2 Pseudocódigo de la Fase de construcción basada en el valor

Procedimiento Búsqueda local

$B \leftarrow 0$
$C = C_1 + C_2$
for all $j \in J$
$c_{max_j}(j) =$ *costo de distribución máximo*
end for all

for all $i \in I$
for all $j \in J$
$\frac{c_{max_j} - c_{ij}}{P_{ij}}$
end for all
end for all

for all $i \in I$
$c_{max_i}(i) = máximo\left(\frac{c_{max_j} - c_{ij}}{P_{ij}}\right)$
end for all

$c_{max_i} \leftarrow sort(c_{max_i})$
$C_1 \leftarrow \left\{ p_i \notin y_k : p_i \text{ seleccionada de los primeros } c_{max_i} \wedge |C_1| = {}^{|C|}/_{2} \right\}$
$C_2 \leftarrow \left\{ p_i \notin y_k : p_i \text{ seleccionada uniformemente al azar} \wedge |C_2| = {}^{|C|}/_{2} \right\}$

for all $(p_r \in y_k \wedge p_i \in C)$
$B \leftarrow \max\{0, (c(p_r) - c(p_i))\}$
if $B > 0$
$Intercambiar\ (p_r, p_i, y_k)$
$Actualizar\ y_k$
return y_k
end if

end for all
end Búsqueda local

Figura A.1.3 Pseudocódigo de la Fase de la búsqueda local

Procedimiento Actualizar PSize

$Población\ (PSize) \leftarrow \infty$
for $i = 1 : |Lista|$
    if $Lista\ (i) < Población\ (PSize)$
        $Población\ (PSize) \leftarrow Lista\ (i)$
    end if

    $Población \leftarrow sort\ (Población)$
end for

end Actualizar PSize

Figura A.1.4 Pseudocódigo de la actualización de la Población

ANEXO 2

TABLAS DE RESULTADOS OBTENIDOS DEL ALGORITMO

A continuación se muestran los resultados obtenidos del algoritmo heurístico Stackelberg-Scatter Search que proponemos para la resolución del problema estudiado en esta tesis. En el algoritmo utilizamos dos parámetros: el grado de aleatoriedad en la lista restringida de candidatos y el número de plantas candidatas que se evalúan con el fin de ingresar a la lista de candidatos. Como mencionamos anteriormente en el capítulo 6, después de pruebas preliminares, los resultados mostraron que un $\alpha = 0.4$ proporciona buenas soluciones en cuestión de calidad y de diversidad y, 6 candidatos parecía ser un valor eficiente para los tres tamaños de todas las instancias.

En las siguientes tablas se muestran los resultados obtenidos de las 10 corridas para cada instancia con los parámetros seleccionados. La primera columna "Iteración" representa el número de corrida de cada instancia, la segunda columna "VFO" indica el valor óptimo de la función objetivo del nivel superior que alcanza la iteración en cada instancia y la tercera columna "Tiempo" muestra el tiempo (en segundos) requerido por el algoritmo para resolver cada instancia. A continuación, de las tablas A.2.1 a A.2.12 son las instancias de tamaño pequeño, es decir, con 50 plantas y 50 clientes.

Iteración	VFO	Tiempo
1	1,137,547.000	2.413
2	1,122,749.500	2.964
3	1,122,749.500	2.589
4	1,159,110.000	2.044
5	1,122,749.500	2.614
6	1,137,547.000	2.817
7	1,122,749.500	2.001
8	1,122,749.500	3.549
9	1,141,293.125	1.716
10	1,167,091.250	2.688

A.2.1 Instancia 132_1 con $p = 8$ y valor óptimo de 1, 122,749.500.

Iteración	VFO	Tiempo
1	1,184,035.625	2.219
2	1,157,722.125	1.965
3	1,192,826.375	2.573
4	1,193,122.000	2.822
5	1,198,406.625	2.29
6	1,198,406.625	1.925
7	1,191,077.250	2.033
8	1,189,615.875	2.057
9	1,204,327.750	2.083
10	1,157,722.125	1.732

A.2.2 Instancia 132_2 con $p = 9$ y valor óptimo de 1, 157,722.125.

Iteración	VFO	Tiempo
1	1,146,301.500	1.568
2	1,189,425.250	1.535
3	1,189,932.500	2.741
4	1,146,301.500	2.482
5	1,196,485.625	2.059
6	1,204,972.250	2.175
7	1,146,301.500	1.908
8	1,146,301.500	1.930
9	1,146,301.500	1.901
10	1,196,485.625	1.864

A.2.3 Instancia 132_3 con $p = 6$ y valor óptimo de 1, 146,301.500.

Iteración	VFO	Tiempo
1	1,036,779.438	3.632
2	1,036,779.438	2.247
3	1,036,779.438	1.530

4	1,036,779.438	1.940
5	1,036,779.438	1.486
6	1,081,489.125	1.768
7	1,036,779.438	1.669
8	1,105,835.125	1.742
9	1,0367,79.438	3.096
10	1,096,963.250	1.890

A.2.4 Instancia 132_4 con $p = 5$ y valor óptimo de 1, 036,779.438.

Iteración	VFO	Tiempo
1	1,126,660.250	2.436
2	1,103,272.000	1.897
3	1,103,272.000	2.647
4	1,195,519.750	2.239
5	1,103,272.000	1.953
6	1,103,272.000	2.064
7	1,126,660.250	2.019
8	1,103,272.000	2.810
9	1,123,272.250	2.432
10	1,103,272.000	2.250

A.2.5 Instancia 133_1 con $p = 7$ y valor óptimo de 1, 103,272.000.

Iteración	VFO	Tiempo
1	1,140,916.500	6.699
2	1,035,443.000	1.992
3	1,035,443.000	2.442
4	1,035,443.000	6.454
5	1,1086,08.000	4.494
6	1,035,443.000	7.743
7	1,074,721.375	1.513
8	1,070,298.000	1.771
9	1,140,916.500	5.437
10	1,035,443.000	1.806

A.2.6 Instancia 133_2 con $p = 5$ y valor óptimo de 1, 035,443.000.

Iteración	VFO	Tiempo
1	1,171,331.250	2.277
2	1,198,878.250	1.702
3	1,198,878.250	2.008
4	1,171,331.250	2.277
5	1,214,743.750	2.013
6	1,171,331.250	1.947
7	1,230,460.875	2.024

8	1,214,743.750	2.033
9	1,230,460.875	1.744
10	1,231,105.000	1.915

A.2.7 Instancia 133_3 con $p = 6$ y valor óptimo de 1, 171,331.250.

Iteración	VFO	Tiempo
1	1,083,636.625	1.991
2	1,087,790.125	2.783
3	1,083,636.625	2.330
4	1,097,835.125	1.659
5	1,083,636.625	1.758
6	1,090,581.875	1.687
7	1,087,790.125	2.285
8	1,087,790.125	1.750
9	1,090,581.875	2.366
10	1,094,106.125	1.764

A.2.8 Instancia 133_4 con $p = 9$ y valor óptimo de 1, 083,636.625.

Iteración	VFO	Tiempo
1	1,179,639.500	2.297
2	1,227,085.500	1.978
3	1,179,639.500	2.483
4	1,179,639.500	2.256
5	1,227,085.500	2.354
6	1,247,996.500	2.062
7	1,179,639.500	2.099
8	1,179,639.500	2.037
9	1,179,639.500	2.337
10	1,1796,39.500	2.256

A.2.9 Instancia 134_1 con $p = 4$ y valor óptimo de 1, 179,639.500.

Iteración	VFO	Tiempo
1	1,153,205.875	1.944
2	1,150,433.250	2.117
3	1,150,433.250	1.801
4	1,121,633.000	2.195
5	1,142,503.625	2.128
6	1,121,633.000	2.757
7	1,163,680.500	1.918
8	1,175,713.250	1.74
9	1,166,220.000	2.563
10	1,121,633.000	1.909

A.2.10 Instancia 134_2 con $p = 7$ y valor óptimo de 1, 121,633.000.

Iteración	VFO	Tiempo
1	1,171,408.500	2.223
2	1,178,007.750	1.914
3	1,171,408.500	1.579
4	1,171,408.500	1.898
5	1,193,109.500	1.574
6	1,195,418.375	2.886
7	1,197,280.500	1.597
8	1,195,418.375	2.255
9	1,195,418.375	1.661
10	1,171,408.500	1.571

A.2.11 Instancia 134_3 con $p = 6$ y valor óptimo de 1, 171,408.500.

Iteración	VFO	Tiempo
1	1,224,143.500	1.612
2	1,210,465.750	1.995
3	1,210,465.750	1.689
4	1,232,325.750	4.747
5	1,210,465.750	5.289
6	1,224,143.500	1.68
7	1,224,143.500	1.813
8	1,210,465.750	4.032
9	1,253,321.750	1.709
10	1,210,465.750	3.175

A.2.12 Instancia 134_4 con $p = 3$ y valor óptimo de 1, 210,465.750.

En las siguientes tablas, de la A.2.13 a la A.2.24, se presentan los resultados de las instancias de tamaño mediado, es decir, de 50 plantas y 75 clientes.

Iteración	VFO	Tiempo
1	1,661,269.500	3.007
2	1,701,684.375	3.334
3	1,661,269.500	7.05
4	1,661,269.500	2.916
5	1,702,240.750	2.956
6	1,706,857.625	3.402
7	1,661,269.500	3.817
8	1,702,240.750	3.771
9	1,661,269.500	2.519
10	1,730,648.125	3.513

A.2.13 Instancia a75_50_1 con $p = 7$ y valor óptimo de 1, 661,269.500.

Iteración	VFO	Tiempo
1	1,632,906.875	5.683
2	1,632,906.875	3.737
3	1,648,421.250	5.458
4	1,632,906.875	3.683
5	1,692,443.125	2.509
6	1,632,906.875	2.735
7	1,632,906.875	2.773
8	1,632,906.875	2.836
9	1,689,849.625	2.828
10	1,632,906.875	3.296

A.2.14 Instancia a75_50_2 con $p = 6$ y valor óptimo de $1,632,906.875$.

Iteración	VFO	Tiempo
1	1,641,094.000	5.417
2	1,632,866.000	3.973
3	1,632,212.500	3.258
4	1,641,094.000	5.490
5	1,639,600.000	2.905
6	1,632,212.500	3.756
7	1,669,354.875	3.637
8	1,632,212.500	3.504
9	1,656,598.625	3.031
10	1,632,212.500	2.939

A.2.15 Instancia a75_50_3 con $p = 7$ y valor óptimo de $1,632,212.500$.

Iteración	VFO	Tiempo
1	1,585,027.625	2.721
2	1,585,027.625	2.831
3	1,585,027.625	2.997
4	1,585,027.625	2.585
5	1,585,027.625	2.943
6	1,585,027.625	2.644
7	1,657,922.750	2.699
8	1,585,027.625	2.244
9	1,585,027.625	2.201
10	1,643,486.625	2.167

A.2.16 Instancia a75_50_4 con $p = 5$ y valor óptimo de $1,585,027.625$.

Iteración	VFO	Tiempo
1	1,346,131.000	4.328
2	1,334,018.625	3.956
3	1,252,803.625	3.125

4	1,252,803.625	3.236
5	1,337,534.750	2.610
6	1,303,203.875	4.531
7	1,252,803.625	2.848
8	1,252,803.625	4.429
9	1,252,803.625	3.067
10	1,252,803.625	4.866

A.2.17 Instancia b75_50_1 con $p = 8$ y valor óptimo de 1, 252,803.625.

Iteración	VFO	Tiempo
1	1,350,387.625	3.324
2	1,349,724.500	4.746
3	1,337,446.375	3.389
4	1,349,209.500	4.427
5	1,341,845.375	3.381
6	1,342,897.750	3.294
7	1,337,446.375	3.445
8	1,337,446.375	4.245
9	1,342,732.000	4.405
10	1,337,446.375	2.992

A.2.18 Instancia b75_50_2 con $p = 9$ y valor óptimo de 1, 337,446.375.

Iteración	VFO	Tiempo
1	1,283,027.375	5.861
2	1,283,027.375	3.963
3	1,272,702.500	4.560
4	1,249,750.250	5.359
5	1,269,546.000	4.766
6	1,290,394.000	3.728
7	1,297,150.000	3.694
8	1,249,750.250	3.837
9	1,297,484.375	4.682
10	1,281,578.000	5.630

A.2.19 Instancia b75_50_3 con $p = 9$ y valor óptimo de 1, 249,750.250.

Iteración	VFO	Tiempo
1	1,217,508.375	3.090
2	1,217,508.375	3.073
3	1,233,503.500	3.091
4	1,217,508.375	3.042
5	1,233,503.500	3.076
6	1,217,508.375	2.984
7	1,217,508.375	2.597

8	1,236,614.250	3.174
9	1,238,170.250	3.862
10	1,217,508.375	2.824

A.2.20 Instancia b75_50_4 con $p = 9$ y valor óptimo de 1, 217,508.375.

Iteración	VFO	Tiempo
1	1,327,507.000	4.662
2	1,335,032.625	5.494
3	1,310,192.875	4.480
4	1,310,192.875	3.466
5	1,318,566.000	3.859
6	1,327,507.000	4.596
7	1,344,231.875	4.793
8	1,327,507.000	3.854
9	1,333,169.375	4.460
10	1,310,192.875	5.065

A.2.21 Instancia c75_50_1 con $p = 11$ y valor óptimo de 1, 310,192.875.

Iteración	VFO	Tiempo
1	1,252,757.625	4.338
2	1,277,413.500	3.271
3	1,244,254.875	4.107
4	1,263,638.750	3.305
5	1,244,254.875	3.155
6	1,249,781.500	4.181
7	1,249,781.500	3.698
8	1,251,792.500	3.367
9	1,276,120.375	3.387
10	1,263,638.750	4.269

A.2.22 Instancia c75_50_2 con $p = 10$ y valor óptimo de 1, 244,254.875.

Iteración	VFO	Tiempo
1	1,241,590.750	4.865
2	1,201,706.500	4.013
3	1,228,039.000	4.541
4	1,294,356.375	4.035
5	1,219,718.125	3.912
6	1,201,706.500	4.108
7	1,258,418.875	4.173
8	1,221,351.000	3.958
9	1,201,706.500	3.487
10	1,201,706.500	4.585

A.2.23 Instancia c75_50_3 con $p = 12$ y valor óptimo de 1, 201,706.500.

Iteración	VFO	Tiempo
1	1,334,782.500	4.915
2	1,348,493.625	4.054
3	1,334,782.500	3.983
4	1,334,782.500	4.024
5	1,347,391.000	4.326
6	1,334,782.500	4.034
7	1,334,782.500	3.620
8	1,342,435.375	3.956
9	1,334,782.500	3.711
10	1,355,136.125	3.782

A.2.24 Instancia c75_50_4 con $p = 11$ y valor óptimo de 1, 334,782.500.

Por último, de la tabla A.2.25 a la tabla A.2.36 son las instancias de 75 plantas y 100 clientes, es decir, las instancias de tamaño grande.

Iteración	VFO	Tiempo
1	2,286,397.500	10.781
2	2,438,037.500	9.493
3	2,286,397.500	10.448
4	2,286,397.500	11.659
5	2,387,755.250	8.992
6	2,321,813.000	13.21
7	2,442,387.750	12.643
8	2,286,397.500	13.044
9	2,286,397.500	9.011
10	2,444,273.750	13.514

A.2.25 Instancia a100_75_1 con $p = 4$ y valor óptimo de 2, 286,397.500.

Iteración	VFO	Tiempo
1	2,463,186.500	11.645
2	2,498,735.250	8.577
3	2,463,186.500	9.964
4	2,463,186.500	12.453
5	2,463,186.500	6.272
6	2,463,186.500	12.111
7	2,520,439.750	12.742
8	2,463,186.500	11.287
9	2,463,186.500	11.121
10	2,509,941.250	7.402

A.2.26 Instancia a100_75_2 con $p = 3$ y valor óptimo de 2, 463,186.500.

Iteración	VFO	Tiempo
1	2,415,836.000	6.132
2	2,422,409.500	10.126
3	2,447,442.250	8.889
4	2,447,442.250	7.669
5	2,447,442.250	6.200
6	2,415,836.000	9.342
7	2,451,884.000	8.202
8	2,415,836.000	5.933
9	2,415,836.000	7.217
10	2,422,409.500	5.954

A.2.27 Instancia a100_75_3 con $p = 3$ y valor óptimo de 2, 415,836.000.

Iteración	VFO	Tiempo
1	2,380,150.250	8.619
2	2,386,980.500	7.092
3	2,451,026.250	6.575
4	2,462,013.250	10.589
5	2,380,150.250	6.327
6	2,449,757.000	8.546
7	2,380,150.250	7.195
8	2,380,150.250	7.549
9	2,380,150.250	9.448
10	2,380,150.250	7.426

A.2.28 Instancia a100_75_4 con $p = 4$ y valor óptimo de 2, 380,150.250.

Iteración	VFO	Tiempo
1	1,998,406.125	19.259
2	1,950,231.375	12.494
3	1,977,686.500	11.604
4	1,996,272.500	18.625
5	1,967,327.125	10.001
6	1,993,992.500	9.906
7	1,996,272.500	8.193
8	1,950,231.375	12.684
9	1,993,992.500	12.459
10	1,996,272.500	19.767

A.2.29 Instancia b100_75_1 con $p = 8$ y valor óptimo de 1, 950,231.375.

Iteración	VFO	Tiempo
1	2,064,564.000	16.531
2	2,064,564.000	8.855
3	2,047,759.500	8.955

4	2,073,979.500	15.766
5	2,064,564.000	20.097
6	2,052,362.375	20.164
7	2,023,097.375	19.543
8	2,065,106.625	13.613
9	2,052,362.375	17.270
10	2,052,074.125	10.740

A.2.30 Instancia b100_75_2 con $p = 8$ y valor óptimo de 2, 023,097.375.

Iteración	VFO	Tiempo
1	2,062,595.000	15.378
2	2,091,992.625	7.086
3	2,091,992.625	9.106
4	2,062,595.000	18.486
5	2,090,019.875	8.706
6	2,090,019.875	10.458
7	2,084,565.375	20.268
8	2,062,595.000	19.945
9	2,062,592.000	20.209
10	2,091,720.625	8.814

A.2.31 Instancia b100_75_3 con $p = 8$ y valor óptimo de 2, 062,595.000.

Iteración	VFO	Tiempo
1	1,906,862.125	7.202
2	1,902,268.625	8.400
3	1,865,323.000	12.935
4	1,924,284.000	13.103
5	1,932,744.250	14.260
6	1,930,856.875	8.134
7	1,902,268.625	13.330
8	1,865,323.000	14.499
9	1,935,221.875	7.073
10	1,930,586.875	11.102

A.2.32 Instancia b100_75_4 con $p = 9$ y valor óptimo de 1, 865,323.000.

Iteración	VFO	Tiempo
1	1,843,620.625	10.002
2	1,874,743.000	7.881
3	1,843,620.625	7.746
4	1,843,620.625	8.936
5	1,871,212.125	13.844
6	1,843,620.625	15.051
7	1,843,620.625	13.326

8	1,843,620.625	12.248
9	1,843,620.625	11.105
10	1,871,212.125	11.271

A.2.33 Instancia c100_75_1 con $p = 6$ y valor óptimo de 1, 843,620.625.

Iteración	VFO	Tiempo
1	1,844,070.625	12.259
2	1,808,867.000	12.869
3	1,822,981.750	14.647
4	1,808,867.000	15.335
5	1,850,725.875	13.111
6	1,846,210.875	13.990
7	1,838,208.500	12.062
8	1,833,753.875	7.371
9	1,855,028.750	14.346
10	1,808,867.000	9.735

A.2.34 Instancia c100_75_2 con $p = 11$ y valor óptimo de 1, 808,867.000.

Iteración	VFO	Tiempo
1	1,839,638.875	9.128
2	1,820,587.375	13.140
3	1,857,770.375	14.547
4	1,820,587.375	9.569
5	1,839,638.875	7.842
6	1,820,587.375	10.428
7	1,839,638.875	9.666
8	1,839,638.875	7.600
9	1,820,587.375	7.794
10	1,820,587.375	15.312

A.2.35 Instancia c100_75_3 con $p = 8$ y valor óptimo de 1, 820,587.375.

Iteración	VFO	Tiempo
1	1,839,007.375	8.568
2	1,851,681.000	10.989
3	1,839,007.375	15.905
4	1,880,863.375	11.722
5	1,881,119.750	14.255
6	1,843,111.875	8.546
7	1,877,902.125	14.404
8	1,839,007.375	12.715
9	1,877,892.250	15.635
10	1,851,681.000	9.700

A.2.36 Instancia c100_75_4 con $p = 9$ y valor óptimo de 1, 839,007.375.