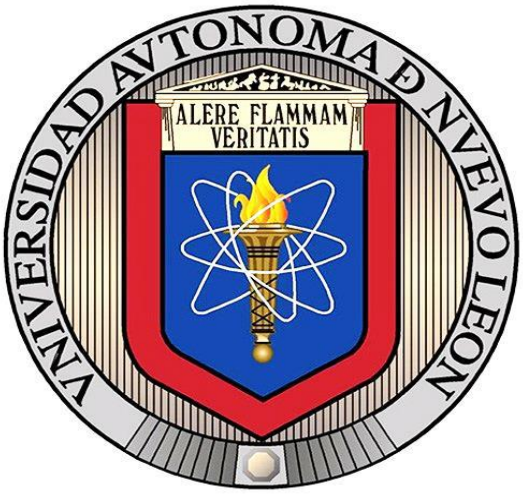
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS



TESIS

ALGORITMO GENÉTICO PARA UN PROBLEMA BINIVEL DEL DISEÑO TOPOLÓGICO DE UNA RED DE ÁREA LOCAL

PRESENTA

M.C. RICARDO PEDRAZA RODRÍGUEZ

PARA OBTENER EL GRADO DE DOCTOR EN CIENCIAS CON ORIENTACIÓN EN MATEMÁTICAS

AGOSTO, 2016

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

CENTRO DE INVESTIGACIÓN EN CIENCIAS FÍSICO MATEMÁTICAS

TESIS

ALGORITMO GENÉTICO PARA UN PROBLEMA BINIVEL DEL DISEÑO TOPOLÓGICO DE UNA RED DE ÁREA LOCAL

PRESENTA

M.C. RICARDO PEDRAZA RODRÍGUEZ

PARA OBTENER EL GRADO DE DOCTOR EN CIENCIAS CON ORIENTACIÓN EN MATEMÁTICAS

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN, MÉXICO AGOSTO DE 2016

ÍNDICE

CAPÍTULO 4. EXPERIMENTACIÓN COMPUTACIONAL

CAPÍTULO 5. CONCLUSIONES Y TRABAJOS FUTUROS

CAPÍTULO 6. REFERENCIAS

CAPÍTULO 1

INTRODUCCIÓN

El diseño topológico de redes de telecomunicación y computadoras es un problema NP-duro de optimización combinatoria. Durante los años recientes, dicho problema ha captado la atención de investigadores y profesionistas debido a su aplicabilidad y uso cotidiano. El problema de diseñar una red de telecomunicaciones consiste en decidir el número, el tipo y la ubicación en la red de todos sus elementos (llámense hubs, switches y routers), también se deberán decidir las conexiones entre sí y sus capacidades. Es evidente que en este problema se pueden plantear varios objetivos de manera jerárquica, los cuales pueden estar en conflicto, tales como el costo monetario, el retardo en la red y el máximo número de hubs. Una vez que se decide el objetivo a considerar se deberá optimizar el problema para alcanzar una solución deseable.

En la literatura existente nosotros podemos identificar dos tipos principales de problemas relacionados con el diseño de redes de telecomunicación: la localización de hubs en Alumur y Kara (2008) y el diseño topológico en Watcharasitthiiwat y Wardkein (2009).El problema de localización de hubs tiene que ver con la decisión de donde localizar algunos hubs en la red y con la asignación de nodos a dichos hubs con el fin de enviar el tráfico entre los pares origen-destino determinado por los usuarios de la red.

Por otro lado, el problema de diseño topológico de una red consiste en la selección de un subconjunto de conexiones que optimizan un criterio de ejecución predeterminado. En esta tesis nos enfocamos en un problema de diseño topológico de una red de área local (LAN por sus siglas en inglés).

Las LANs son comúnmente usadas como infraestructuras de comunicación que satisfacen la demanda de un conjunto de usuarios en un ambiente local. Usualmente

estas redes se forman por varios segmentos de LAN conectados por puentes. El problema del diseño topológico de una LAN consiste en encontrar la mejor configuración entre usuarios y clusters que optimice uno (o varios) criterios de desempeño previamente definidos, tales como, costo de equipamiento, costo de conexión, tiempo de respuesta, fiabilidad de la red, entre otros (ver Ersoy y Panwar, 1993). Estos criterios de rendimiento son muy importantes y son afectados de manera significativa afectados por la topología de red, es decir, por la forma en que están hechas las conexiones.

El problema del diseño óptimo de la red es un problema combinatorio difícil que involucra decisiones de asignación y enrutamiento. En el problema de asignación se determina la mejor manera de asignar usuarios a clusters; mientras que en el problema de enrutamiento se determinan los segmentos donde los clusters necesitan ser interconectados a través de puentes de tal manera que formen un árbol de expansión mínima. Típicamente el diseño de redes de comunicación requiere la existencia de un árbol de expansión, en el cual cada nodo debe estar comunicado con todos los demás. Sin embargo, en Estepa y Estepa (2011) remarcan que las soluciones de un árbol de expansión no proveen un diseño fiable para un diseño de costo mínimo ya que al eliminar una conexión se va a tener una red disconexa, lo cual no sucede con una estructura de conexión en forma de ciclo. Pero desde el punto de vista económico, tener un ciclo genera un costo mayor que tener un árbol.

1.1 REVISION DE LITERATURA

El diseño topológico de LANs ha sido un área de investigación muy activa en las últimas dos décadas. Muchos autores han propuesto métodos exactos para resolverlos (por ejemplo en Gavish (1971), Rothlauf (2009), y Mateus *et al*. (2001)), mientras que otros han propuesto técnicas heurísticas (Glover y Lee, 1991, Jan *et al.,* 1993, Chu y Premkumar, 2000, Estepa y Estepa, 2011 y Khan y Engelbrecht, 2012) para diseñar las

LANs. Cabe mencionar que los algoritmos genéticos han tenido una marcada preferencia sobre otras metaheurísticas.

Por otro lado, de la misma forma que los problemas de LANs, en los últimos 25 años el campo de la optimización binivel ha recibido considerable atención, lo cual se ve reflejado en una amplia variedad de aplicaciones. Debido a la complejidad existente en resolver problemas de programación binivel, los algoritmos metaheurísticos han sido considerados como una buena alternativa para encontrar soluciones de alta calidad a problemas binivel. Hay trabajos en los campos de estudios ambientales (ver Kara y Verter, 2004), Logística Humanitaria (ver Jing *et al*., 2010 y Feng y Wen, 2005), diseño de redes (ver LeBlanc, 1973 y Marcotte *et al*., 2009), transporte (ver Yang y Bell, 2001 y Gallo *et al*., 2010), fijación de cuotas de peaje (ver Labbé et al., 1998 y Kalashnikov *et al.,* 2010), localización (Hansen *et al.,* 2004 y Eiselt y Laporte, 1996), planificación de la producción (ver Calvete *et al.*, 2011), entre muchos otros. Para una descripción detallada de más aplicaciones y métodos de solución remitimos al lector a Vicente y Calamai (1994) y Colson *et al.* (2007).

A continuación, se resumen algunos trabajos sobre metaheurísticas diseñadas para resolver problemas binivel.

Trabajos previos relevantes

Al mejor de nuestro conocimiento Mathieu *et al.* (1994) presentan la primera aplicación en la literatura de algoritmos genéticos para resolver problemas de programación binivel. En este trabajo, la población inicial de soluciones corresponde a valores para la solución del líder y las respuestas del seguidor fueron obtenidas directamente optimizando el problema lineal del nivel inferior. Para cada solución del líder el problema del seguidor es resuelto de manera exacta.

Luego, en Oduguwa y Roy (2002) se proporciona un esquema general de un algoritmo genético capaz de resolver diferentes aplicaciones modeladas con programación binivel. Ellos plantean un método con dos poblaciones, una para el líder y otra correspondiente al seguidor. Asumen que el seguidor coopera con el líder después de optimizar el nivel inferior. La cooperación se realiza al final de cada iteración, es decir, después de un ciclo de evolución. Se hace una sincronización (intercambio) entre las poblaciones del líder y el seguidor con el fin de preservar la naturaleza interactiva del problema.

Bhadury *et al*. (2002) proponen diversas variantes de algoritmos genéticos para resolver una amplia variedad de problemas de localización, incluyendo problemas de localización de plantas competitivos. Con el fin de resolver el problema de nivel inferior ellos diseñaron una heurística voraz, la cual fue implementada cada vez que se generaba una nueva solución del líder.

También Yang *et al.* (2009) proponen un algoritmo híbrido que combina el método simplex, algoritmos genéticos y simulaciones difusas y estocásticas para resolver un problema de binivel de localización donde el flujo de información del nivel inferior al nivel superior es estocástico. Para cada solución del líder el nivel inferior es resuelto de manera exacta.

En Alekseeva *et al.* (2009) se propone un algoritmo híbrido que combina los principios de los algoritmos evolutivos con la búsqueda tabú para un problema competitivo de la p-Mediana. Para hacer una evaluación de la solución del líder ellos resuelven el problema del nivel inferior a través de un software comercial de optimización.

Continuando con la misma línea de investigación, en Vasilyev y Klimentova (2009) diseñaron un algoritmo híbrido que combina el método de recocido simulado con el método de ramificación y acotamiento, con el fin de obtener cotas superiores para el problema de localización de instalaciones con preferencias de los clientes. Durante el

método de recocido simulado ellos resuelven el problema del nivel inferior para cada solución del líder.

También hay trabajos en donde el nivel inferior no se resuelve a optimalidad para cada solución del líder, tal como lo propone Gallo *et al.* (2010). Ellos proponen un algoritmo de búsqueda dispersa para resolver un problema de diseño de red de transporte urbano. El nivel superior puede ser visto como un modelo para resolver el problema de diseño topológico de red. Mientras que el modelo del nivel inferior tiene como objetivo resolver el problema de ajuste de la señal. Con el fin de evitar la necesidad de resolver el nivel inferior en cada iteración, es decir, el problema de ajuste de la señal, ellos proponen un enfoque local. En consecuencia, el problema de ajuste de la señal fue formulado como un problema de asignación con equilibrio asimétrico, donde sólo las variables del diseño topológico asumen el papel de variables de decisión. Entonces, ambas decisiones, el ajuste de la señal y el equilibrio entre los flujos de tráfico, son variables descriptivas reduciendo el problema binivel en uno de un solo nivel. El lado negativo es que debido al tamaño de los vecindarios, ellos usaron un método de búsqueda aleatoria para mejorar las soluciones.

Luego, en Gallo *et al.* (2010) se diseñó un algoritmo de búsqueda tabú para resolver un problema binivel de localización de plantas competitivas. Ambos, el líder y el seguidor, buscan maximizar sus propios beneficios considerando las instalaciones ya existentes y las nuevas abiertas por la compañía de la competencia. El nivel inferior es resuelto mediante un algoritmo de ramificación y acotamiento aplicado a una relajación del programa no lineal resultante debido a las propuestas introducidas por ellos.

En Calvete *et al.* (2011) se propone un algoritmo de optimización basado en colonias de hormigas para resolver un problema de programación binivel de la producción y distribución, donde el nivel superior consiste en resolver un problema de ruteo de vehículos con múltiples depósitos, y el modelo de nivel inferior resultante resuelve el problema de la minimizar los costos de manufactura. En el algoritmo binivel de

optimización mediante colonia de hormigas, el problema del nivel inferior se resuelve de manera exacta para cada solución del líder.

Dentro de los métodos con dos poblaciones, una para cada nivel de decisión del problema podemos considerar a Legillon *et al.* (2012). En ese trabajo se diseñó un algoritmo co-evolutivo para resolver el mismo problema binivel de planeación de producción y distribución. Como ya se mencionó, ellos proponen dos poblaciones iniciales, una para el líder y otra para el seguidor las cuales intercambian información periódicamente para mejorar los individuos de dichas poblaciones. Los individuos son creados como la unión de las soluciones del líder y del seguidor. La función de aptitud es evaluada basada en la función objetivo del líder.

En Brotcorne *et al.* (2012) se diseñó un algoritmo de búsqueda tabú para resolver el problema binivel de fijación de cuotas de peaje en una red de transporte. El líder quiere maximizar el beneficio de los peajes obtenidos de la red de transporte, mientras que el seguidor busca minimizar el costo total de su viaje. Con el fin de obtener la respuesta óptima del seguidor, se considera una reformulación del nivel inferior, entonces, aplican el método de generación de columnas y resuelven el problema resultante por optimización inversa.

Por último, Camacho-Vallejo *et al.* (2014) proponen un algoritmo Stackelberg-Evolutivo para resolver un problema binivel de localización de plantas considerando las preferencias de los clientes. El nivel superior busca minimizar los costos de ubicación y distribución mientras que el seguidor intenta minimizar una función utilidad basado en las preferencias de los clientes. En cada iteración del algoritmo propuesto una solución del líder es obtenida, entonces ésta se le provee al seguidor, quien optimiza el problema de asignación del nivel inferior con el objetivo de obtener la respuesta óptima, y así, poder evaluar la función objetivo del nivel superior objetivo.

De los trabajos recién descritos se puede observar que en la mayoría de los trabajos previos se consideran metaheurísticas para resolver a los problemas de optimización binivel. En algunos de ellos el problema del seguidor es resuelto de manera exacta. La principal dificultad derivada del problema considerado en esta tesis es que el problema del seguidor es un problema combinatorio difícil. Por lo tanto, no se puede resolver de manera óptima en un tiempo computacional razonable. Además, hay que considerar la gran cantidad de veces que el problema del seguidor necesita ser resuelto. Aquí es donde la reacción racional del seguidor se toma en cuenta, es decir, dicha reacción se realiza en sentido de no resolver a optimalidad su problema y conformarse con una solución de buena calidad.

Por otra parte, desde un punto de vista de teoría de juegos, los problemas de dos jugadores pueden ser abordados desde el enfoque de Nash, Camacho-Vallejo *et al.* (2014). Ambos enfoques han sido estudiados de manera amplia en la literatura. La existencia de un líder y un seguidor asemeja al problema aquí analizado a los juegos de Stackelberg. Un juego de Stackelberg se compone de un vector de variables de decisión y ara el líder en el nivel superior, y un vector de variables de decisión x ara el seguidor en el nivel inferior. Se supone que al líder se le otorga la primera opción de elegir y selecciona una solución y de acuerdo con sus restricciones con el fin de optimizar su función objetivo; ésta decisión se toma teniendo en cuenta la reacción racional del seguidor. Teniendo en cuenta la decisión del líder, el seguidor selecciona una solución factible $x(y)$ para él con el objetivo de optimizar su propia función objetivo; esto es, la reacción del seguidor depende de la decisión tomada por el líder. Por otro lado, el equilibrio de Nash ocurre cuando múltiples jugadores toman una decisión de manera simultánea en el mismo nivel de jerarquía considerando las decisiones de los otros competidores como fijas. Entonces, cualquier jugador puede tomar en cuenta la posibilidad de cambios respecto a las estrategias de los otros jugadores. Por lo tanto, el equilibrio Nash puede ser apropiado aplicarlo en problemas de programación multiobjetivo.

Para el caso de los problemas de programación binivel, el enfoque que parece ser más apropiado es el de encontrar el equilibrio de Stackelberg, el cual considera la existencia de una jerarquía predefinida entre los jugadores. En primer lugar, el líder hace su decisión y basada en dicha decisión, el seguidor elige esta decisión y el líder conoce exactamente la decisión del seguidor. Por lo tanto, el líder tiene la posibilidad de tener en cuenta la respuesta óptima del otro jugador. En Kim *et al.* (2009) los autores justifican su enfoque propuesto asumiendo que el problema binivel puede ser modelado como un juego de Nash si los jugadores tratan de optimizar su propio beneficio en una manera no cooperativa.

Sin embargo, reducir una solución del programa binivel al concepto de equilibrio de Nash clásico no puede hacerse de una manera simple y directa. Para esto, remitimos al lector a Stackelberg (1934), Wang y Periaux (2001) y Pieume *et al.* (2009). Primero, Wang y Periaux (2001) estudiaron un problema bi-objetivo, donde compararon tanto el enfoque Nash como el Stackelberg. En ambos casos emplearon un algoritmo genético, pero para poder estudiar el caso de Stackelberg, los autores definen un líder y un seguidor para cada esquema experimental. Los resultados experimentales muestran que el enfoque Stackelberg logra mejores resultados (aunque es más costoso computacionalmente). Concluyen que Nash y Stackelberg son significativamente diferentes y el enfoque adecuado depende del problema particular.

Además, Pieume *et al.* (2009) adaptan una técnica de optimización multiobjetivo para resolver una clase particular de problemas de programación binivel en donde la solución óptima binivel está determinada por los puntos óptimos de Pareto que corresponden a los puntos no dominados que pertenecen a la intersección de los dos conjuntos eficientes. Sin embargo, con el fin de encontrar los puntos eficientes, los conos deben ser convexos y en muchos casos no lo son. Por otra parte, la metodología propuesta se basa en ir solucionando dos problemas bi-objetivo intercambiando el líder y el seguidor de forma independiente. Esta experimentación se realiza con el fin de validar la relación existente entre ellos. Ellos concluyen que las técnicas multi-criterio

no han demostrado ser buena opción para resolver problemas binivel. Además, Calvete y Galé (2010) estudiaron las diferencias entre problemas de programación bi-objetivo y bi-nivel. Los autores observaron que ha habido intentos de establecer una relación entre ambos tipos de problemas y no se ha llegado a un acuerdo formal. Por otra parte, varios contraejemplos que niegan cualquier relación entre ambos enfoques pueden ser encontrados en la literatura (por ejemplo, Candler, 1988, Clarke y Westerberg, 1988, Haurie *et al.,* 1990 y Marcotte y Savard, 1991).En esos trabajos, los autores muestran empíricamente (y gráficamente) que las soluciones óptimas de un problema binivel no se encuentran dentro la frontera óptima Pareto de un problema bi-objetivo.

1.2 DESCRIPCION DEL PROBLEMA

El problema considerado en esta tesis puede describirse como sigue: se desea realizar el diseño topológico de una red de área local planteado como un problema de programación binivel en el cual hay un líder que quiere minimizar el costo total de conexión considerando que cada usuario debe estar conectado a un solo clúster. Por otra parte, existe un seguidor que quiere minimizar el tiempo promedio de retraso del mensaje sabiendo que al formarse un árbol cubriente, los puentes que conectan clústeres con clústeres no deben de formar ciclos y no pueden rebasar la capacidad disponible con respecto al tráfico que fluye a un clúster determinado.

1.3 OBJETIVO

El objetivo de esta tesis consiste en estudiar el problema del diseño topológico de una LAN y proponer una metodología de solución que proporcione soluciones de buena calidad a un bajo costo computacional.

El algoritmo propuesto es una buena alternativa para resolver el problema binivel; el método empleado es un Algoritmo Genético que considera el equilibrio de Stackelberg

durante el proceso. Es decir, para cada solución del líder se encuentra la reacción racional del seguidor.

El otro objetivo es validar la eficiencia del algoritmo propuesto basándose en un análisis de los resultados obtenidos de la experimentación computacional. Por último, también se discutirá sobre la diferencia existente en la solución de problemas binivel mediante un enfoque de equilibrio de Stackelberg o de Nash.

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 sobre los dos tipos principales de problemas relacionados con el diseño de redes de telecomunicación.
3. Revisión de literatura del problema del diseño topológico de redes LAN.
4. Revisión de literatura sobre problemas binivel de diseños de redes LAN.
5. Revisión de literatura de algoritmos metaheurísticos aplicados para resolver aplicaciones de problemas modelados con programación binivel.
6. Examinar el esquema de los Algoritmos Genéticos para definir si es buena alternativa para la resolución del problema.
7. Análisis las propiedades del problema así como el planteamiento del modelo matemático.
8. Familiarización del lenguaje C++.
9. Desarrollo e implementación computacional del algoritmo propuesto así como su evaluación computacional.
10. Generación y adaptación de un conjunto de instancias para probar el desempeño del algoritmo propuesto.
11. Experimentación preliminar para afinar los parámetros del Algoritmo Genético, sustentado por análisis estadístico.

12. Experimentación computacional con el algoritmo afinado.
13. Presentación de los avances de tesis en el XVII Congreso Latino Americano de Investigación de Operaciones, celebrado en Monterrey, México en Octubre de 2014.
14. Análisis estadístico y gráfico para inferir conclusiones sobre la eficiencia del algoritmo.
15. Escritura de un artículo sometido a la revista arbitrada Celerinet, derivado de una parte de la investigación realizada en esta tesis.
16. Escritura de un artículo sometido a la revista indexada PLoS ONE, cuyo contenido fueron los resultados completos de esta tesis.
17. Escritura y revisión de la tesis.

1.5 ESTRUCTURA DE LA TESIS

En este primer capítulo se vio la descripción de problema del diseño topológico de una red LAN y se dio a conocer el objetivo principal de la tesis. También se describieron las contribuciones en el desarrollo de metaheurísticas para solución de problemas binivel relacionados. Por último se detalla la metodología propuesta para la obtención del producto final, es decir, de la tesis.

El capítulo 2 presenta la definición de los parámetros y variables inmiscuidas en el modelo matemático y se muestra la formulación bi-nivel del problema. Después en el capítulo 3 se describe el método de solución propuesto para la obtención de soluciones bi-nivel de alta calidad. Se presenta una breve introducción a los algoritmos genéticos y se detalla cada componente involucrado en el algoritmo.

Luego, en el capítulo 4 se muestran los resultados obtenidos de los experimentos computacionales llevados a cabo sobre el conjunto de las instancias reportadas. Se muestran los resultados preliminares para la afinación de parámetros y los resultados

con el algoritmo afinado. Para dar mayor validez al algoritmo propuesto, se muestran gráficas y tablas de resultados con su respectivo análisis estadístico.

Por último, en el capítulo 5 se dan las conclusiones asociadas remarcando la importancia de resolver los problemas bi-nivel con la metodología apropiada, es decir, el enfoque de Stackelberg en lugar del enfoque de Nash; y se presentan algunas posibles extensiones de trabajo futuro.

CAPÍTULO 2
MODELACION DEL PROBLEMA

En este capítulo se va a presentar la descripción del problema bajo estudio en esta tesis. Además se va a formular matemáticamente dicho problema y por último se va a hacer una breve discusión acerca de la complejidad que presenta este problema afectando así las técnicas de resolución.

2.1 DESCRIPCION DEL PROBLEMA

Un problema de programación bi-nivel es un problema de programación matemática el cual está compuesto por una función objetivo en el nivel superior y dentro de sus restricciones, otra función objetivo con sus propias restricciones en el nivel inferior. En este trabajo, el problema del nivel superior tiene como objetivo minimizar el costo de conexión, mientras que el problema del nivel inferior busca minimizar el tiempo promedio de retardo de los mensajes. Cuando resolvemos el problema bi-nivel, tanto el tomador de decisiones del nivel superior (en adelante, el líder) y el tomador de decisiones del nivel inferior (en adelante, el seguidor) interactúan jerárquicamente para conseguir la mejor solución.

En el problema aquí considerado se va optimizar el costo de conexión considerando que cada usuario deberá estar conectado a un solo clúster y que las conexiones entre clusters no deben rebasar su máxima capacidad de información que fluye a un clúster determinado teniendo en cuenta que no se deben formar ciclos en el árbol tal que el tiempo promedio de retraso del mensaje se minimice. De esta manera, el líder debe decidir la forma en que será la conexión de usuarios a clusters y el seguidor después debe decidir la forma de conectar a un clúster con otro clúster.

2.2 FORMULACION MATEMATICA

A continuación se va a presentar una definición formal del diseño del problema bi-nivel de una red de área local (BLANDP), sea $N = \{1,2,\dots,n\}$ el conjunto de usuarios (e.g. routers) en la red de telecomunicaciones, y sea $G = (V, E)$ un grafo no dirigido, donde $V = \{v_1, v_2, \dots, v_m\}$ es el conjunto de vértices (*clusters*) y $E = \{(v_p, v_q): p < q\}$ el conjunto de aristas (*puentes*) que conectan los clusters. Para cada clúster, la capacidad máxima de tráfico C_p que puede fluir a través de este es conocida. También, para cada puente el tiempo de respuesta promedio b_{pq} para enrutar un paquete de entre los respectivos clusters es conocida. Asumimos que las características de tráfico entre usuarios son conocidas y se resumen en la matriz de tráfico entre usuarios S, donde un elemento $s_{ij} \in S$ representa el tráfico desde el usuario $i \in N$ al usuario $j \in N$.

Dos costos fundamentales son considerados en el problema binivel: el costo de conexión entre clusters $\{w_{pq}: (p, q) \in E\}$ y el costo de conexión entre usuarios y clusters $\{\alpha_{ip}: i \in N, p \in V\}$.

Consideramos las siguientes variables de decisión:

Para el líder

$$y_{ip} = \begin{cases} 1, & \text{si el usuario } i \text{ es asignado al clúster } p. \\ 0, & \text{de otra manera.} \end{cases}$$

Mientras que para el seguidor

$$x_{pq} = \begin{cases} 1, & \text{si el clúster } p \text{ es conectado al clúster } q. \\ 0, & \text{de otra manera.} \end{cases}$$

La variable de decisión x_{pq} está definida como $x \in X$, donde X es un conjunto de árboles de expansión. Para estas variables de decisión otros elementos importantes relacionados al tráfico en la red serán definidos. Para definir estos términos es necesario la introducción de los siguientes conceptos. Una ruta entre $0 \in V$ and $r \in V$, i.e., ruta$(0, r)$, es una secuencia de vértices sin repetición $(v_{i-1}, v_i) \in E$ para toda $i =$

$1, \ldots, r$. Un vértice v_k es llamado vértice intermedio en el $camino(0, r), si\ \{v_0, \ldots, v_k, \ldots, v_r\}$. Similarmente, nosotros definimos el concepto de arista intermedia como el conjunto de todas las aristas (p, q)en $el\ camino(0, r)$. Con estas definiciones y los conceptos anteriores nosotros precisamos los siguientes términos que pueden ser vistos como variables auxiliares, sea:

Γ	Es el tráfico total que se ofrece en la red, el cual puede ser calculado como $\Gamma = \sum_{i=1}^{N} \sum_{j=1}^{N} s_{ij}$ o por $\Gamma = \sum_{p \in V} \sum_{q \in V} t_{pq}$.
T	Es la matriz de tráfico entre clusters, la cual puede ser calculada como $T = Y^T S Y$, donde Y es la matriz de clusters, la cual asigna usuarios a clusters. Un elemento t_{pq} de esta matriz representa el tráfico enviado desde el clúster $p \in V$ al clúster $q \in V$.
$L(x)_k$	Es el tráfico total al clúster $k \in V$, este puede ser calculado como $$L(x)_k = \sum_{p \in V} t_{pk} + \sum_{q \in V \setminus \{k\}} t_{kq} + \sum_{\{p,q \in V \mid k \in path(p,q)\}} t_{pq} \quad (1)$$
$F(x)^{(p,q)}$	Sea el tráfico total el cual fluye sobre el puente $(p, q) \in E' \subseteq E$, calculado como $$F(x)^{(p,q)} = \sum_{\{k,r \in V \mid (p,q) \in path(k,r)\}} t_{kr} \quad (2)$$

El problema de optimización del líder consiste en determinar la mejor ubicación de usuarios a clusters de tal manera que los costos de conexión se minimicen. Por otra parte, el problema de optimización del seguidor es determinar el subconjunto de aristas $E' \subseteq E$ que formen un árbol de expansión $T = (V, E')$ en G que minimiza el tiempo promedio de retardo de los mensajes en la red.

Cabe señalar que las variables auxiliares $L(x)_k$ y $F(x)^{(p,q)}$ dependerán en gran medida de la configuración de la red definida por el árbol de expansión x. A continuación se

muestra una tabla con el listado de todos los conjuntos, parámetros, variables de decisión y variables auxiliares del problema.

Conjuntos	
N	Número de usuarios(routers)
G	un grafo no dirigido
V	Número de vértices(clusters)
E	Número de aristas
Parámetros	
C_p	Capacidad máxima de tráfico en un clúster
b_{pq}	Tiempo de respuesta promedio para enrutar un paquete de entre los respectivos clusters
w_{pq}	Costo de conexión entre clusters
α_{ip}	Costo de conexión entre usuarios y clusters
Variables de decisión	
y_{ip}	Si el usuario i es asignado al clúster p
x_{pq}	Si el clúster p es conectado al clúster q
Variables auxiliares	
Γ	Es el tráfico total que se ofrece en la red
T	Es la matriz de tráfico entre clusters
$L(x)_k$	Es el tráfico total al clúster $k \in V$
$F(x)^{(p,q)}$	Sea el tráfico total el cual fluye sobre el puente $(p,q) \in E' \subseteq E$

Tabla 2.1. Definición de componentes involucrados en el modelo matemático

El modelo matemático bi-nivel del problema considerado está dado por:

Bi-level Local Area Network Design Problem (*BLANDP*)

$$\min_{y} \sum_{(p,q)\in E} w_{pq}x_{pq} + \sum_{i=1}^{n}\sum_{p\in V} \alpha_{ip}y_{ip} \tag{3}$$

Sujeto a:
$$\sum_{p\in V} y_{ip} = 1 \qquad \forall i = \{1, \dots, n\} \tag{4}$$

$$y_{ip} \in \{0,1\} \qquad \forall i = \{1, \dots, n\}, p \in V \tag{5}$$

donde x resuleve

$$\min_{x} \ \frac{1}{\Gamma}\left[\sum_{p\in V} \frac{L(x)_p}{c_p - L(x)_p} + \sum_{p\in V}\sum_{q\in V} F(x)^{(p,q)} b_{pq}\right] \tag{6}$$

Sujeto a:
$$\sum_{(p,q)\in A} x_{pq} = m - 1 \tag{7}$$

$$\sum_{(p,q)\in(S,S)} x_{pq} \leq |S| - 1 \qquad \forall S \subseteq V \tag{8}$$

$$L(x)_p < C_p \qquad \forall p \in V \tag{9}$$

$$x_{pq} \in \{0,1\} \qquad \forall p, q \in V \tag{10}$$

La función objetivo en la ecuación (3) minimiza el costo total de conexión (el objetivo del líder). El primer término se refiere al costo de conexión entre clusters determinado por el árbol de expansión, mientras el segundo término se refiere al costo de ubicación entre usuarios y clusters. La ecuación (4) establece que cada usuario puede estar conectado únicamente a un solo clúster. La restricción (5) indica que las variables del líder son binarias.

La función objetivo del seguidor dado en (6) minimiza el tiempo de retardo promedio en los mensajes. El retardo total promedio en la LAN está compuesta de los retardos de los clusters y los puentes (ver Elbaum y Sidi, 1996), y esto da como resultado una función no lineal. Cabe señalar que un árbol debe satisfacer las siguientes condiciones de árboles: tiene que tener solamente $m - 1$ aristas, todos sus vértices deben estar

conectados y no deben existir ciclos. Es bien sabido que dos de estas condiciones implican la tercera. Por lo tanto la ecuación (7) establece que un árbol debe tener exactamente $m - 1$ aristas, mientas la ecuación (8) hace cumplir la restricción de que las aristas de T no pueden formar ciclos, donde (S, S) denota todas las aristas que van desde un vértice en el conjunto S a otro vértice en el conjunto S. Ambas ecuaciones implican un árbol de expansión. La restricción (9) establece una condición de capacidad para el tráfico que fluye a través de un clúster. Finalmente, las restricciones (10) indican la naturaleza binaria de las variables de decisión del seguidor.

2.3 DISCUSIÓN DE LA COMPLEJIDAD

Para asegurar que el problema binivel denotado por (3)-(10) está bien definido, asumimos que si el seguidor tiene múltiples respuestas óptimas para cualquier decisión del líder, entonces la decisión del seguidor que sea más conveniente para el líder será seleccionada. Este caso se conoce como la versión optimista del problema binivel.

El hecho de que el nivel inferior sea un problema que no se puede resolver a optimalidad de manera sencilla o con bajo esfuerzo computacional nos orilla a buscar métodos alternativos para resolver el nivel inferior con soluciones cercanas a las óptimas. Esto se discutirá y mostrará en el siguiente capítulo.

CAPÍTULO 3

METODOLOGÍA DE SOLUCIÓN

En este capítulo se va a dar una idea general sobre los algoritmos metaheurísticos y se describirán algunos de ellos. En particular, vamos a definir claramente todos los componentes involucrados en los algoritmos genéticos debido a que fue la técnica seleccionada para proponer nuestro algoritmo de solución.

También, se va a describir el algoritmo propuesto en esta tesis y se explicará la forma en que se codificaron las soluciones en nuestro algoritmo.

3.1 ALGORITMOS METAHEURISTICOS

Antes de comenzar con los algoritmos metaheurísticos es conveniente explicar el término "heurística" que proviene de la palabra griega *euriskein* que significa hallar o encontrar, la cual la utilizó por primera vez Arquímedes cuando descubrió el principio que lleva su nombre.

En el campo de la optimización matemática y la investigación de operaciones el término heurístico es un procedimiento para el cual se tiene un alto grado de confianza en encontrar soluciones de alta calidad con un costo computacional razonable, aunque no se garantice la optimalidad o factibilidad de dichas soluciones.

El término "metaheurística" significa algo que va más allá de una heurística y apareció por primera vez en un artículo sobre búsqueda tabú de Fred Glover en el año de 1986 (ver Moreno ,2004).

Dentro de los Factores que pueden hacer interesante el uso de las metaheurísticas son:

a) Cuando no hay un método exacto de resolución, o éste requiere mucho tiempo de cálculo y memoria (es decir, que sea ineficiente)
b) Cuando no se necesita la solución óptima, es decir, que basta con una solución de buena calidad

Las metaheurísticas son estrategias generales de diseño de procedimientos heurísticos para la resolución de problemas de alto rendimiento.

Tal como se clasifica en Moreno (2004), los tipos de metaheurísticas se establecen en función de las características en los procedimientos que intervienen en el método:

a) Métodos de relajación
b) Procesos constructivos
c) Búsquedas por entornos
d) Procedimientos evolutivos

De manera análoga, a continuación se muestra un esquema de clasificación por tipos de metaheurísticas.

1.- Relajación (Lagrangiana, Restricciones surrogadas, entre otros)

- Proponer soluciones
- Guiar la búsqueda

2.- Constructivas

- Voraz
- Aleatoria
- GRASP

3.- Búsqueda

- Búsqueda Local
- Búsqueda Global
    - MultiStart (MS)
    - Vecindades Variables (VNS)
    - Memoria (Tabu Search)
    - Estocásticas (Recocido Simulado)

4.- Evolutivas

- Evolución estocástica (Algoritmos genéticos, Meméticos, Evolución diferencial)
- Evolución determinística (Scatter Search, Path-Relinking)

5.- Otros Tipos

- Descomposición
- Memoria a largo plazo

Como se puede observar, existen muchos tipos y clasificaciones de las metaheurísticas, en esta tesis utilizamos una metaheurística evolutiva estocástica llamada Algoritmos Genéticos (GA, por sus siglas en inglés). Es por esto que ahora se van a describir todos los componentes inmersos en un algoritmo genético.

3.2 ALGORITMOS GENETICOS

En esta subsección, se describen los componentes de cualquier algoritmo genético genérico. Existe una similitud entre sus componentes y los que participan en la evolución de las especies, de ahí el nombre que tienen. Los GA operan sobre un conjunto de individuos (soluciones) que forman una población en una determinada generación (iteración), entonces dos individuos –los padres- son seleccionados y combinados en un operador genético que simule la evolución. La mayoría de las veces dichos operadores consisten en una operación de cruzamiento y en una fase de mutación. Estos cruzamientos y las mutaciones son realizados aleatoriamente con el fin de generar hijos (nuevas soluciones). Luego, basado sobre un criterio de selección, los

individuos más fuertes (los que tienen un mejor valor en la función objetivo) sobreviven y permanecen para la siguiente generación. El proceso se repite hasta que algunas condiciones de paro se cumplan. Un marco general para GA se muestra en la figura 1. Para poder resolver el BLANDP, una adaptación adecuada de los diferentes componentes del GA necesita hacerse; más adelante dichos componentes serán descritos.

Generar una población P_k

Evaluar la aptitud de la población P_k

while ($k \leq Max\_Generations$)

    Seleccionar algunos individuos de P_k

    Cruzamiento de algunos individuos seleccionados y colocar a los hijos en P'_k

    Mutar algunos de los individuos seleccionados y coloca la mutación en P'_k

    Evaluar a los individuos en P'_k

    Crear P_{k+1} seleccionando algunos individuos desde $P_k \cup P'_k$

    $k = k + 1$

end while

return el individuo desde $P_{Max\_Generations}$ con mejor valor de aptitud

Figura 1. Esquema de un Algoritmo Genético.

3.3 JUSTIFICACIÓN DEL ALGORITMO

Los problemas de programación binivel generalmente son difíciles de resolver porque la evaluación de la función objetivo del nivel superior requiere la solución del problema de optimización nivel inferior. Más aún, puesto que el nivel inferior considerado en este trabajo consiste en problema de programación no lineal, el problema binivel resulta en un problema de programación no convexa. El problema del nivel inferior tiene como objetivo de minimizar el problema de retraso promedio de los mensajes en un árbol cubriente. Es decir, en cada paso del proceso de optimización del problema de nivel

superior, que es básicamente un problema de asignación, se tiene que hallar el mejor árbol cubriente. Por lo tanto, se propone un GA para resolver el problema considerado en esta tesis.

La motivación que tuvimos para tomar la decisión para usar un GA en esta aplicación en particular se basa en el hecho de que es una técnica muy flexible que puede ser adaptada de varias maneras a muchos problemas de optimización definiendo adecuadamente los criterios utilizados en los operadores genéticos definidos dentro del procedimiento de solución. Los GA usan estrategias para la diversificación manteniendo la mayor parte de las soluciones de buena calidad; y han demostrado ser eficaces en la resolución de problemas de tipo multi-objetivo, de optimización robusta y de programación binivel. Dado que los GA son metaheurísticas basadas en una población, se requiere que los cromosomas que representan una solución sean eficientes. Antes de describir la implementación del GA propuesto, definimos la codificación solución (cromosoma) y la evaluación de la función objetivo (función de aptitud).

3.4 CODIFICACION DE LA SOLUCIÓN

El propósito de nuestro problema es agrupar usuarios y asignarlos a algún clúster. Luego, checar que los clusters que forman el árbol de expansión no excedan las restricciones de capacidad en los puentes (conexiones entre sí). Por lo tanto, la estructura más simple para representar una solución factible puede ser en un arreglo $y = \langle y(1), y(2), \dots, y(n) \rangle$, donde n es el número de usuarios en la red. Cada posición de y indica el clúster p al cual el i-ésimo usuario ha sido asignado ($y(i) = p$). Una solución factible debe satisfacer el requisito que cada usuario se debe asignar a un solo clúster, el cual es obviamente satisfecho con esta representación. Por otro lado, la configuración del árbol de expansión es representada por una lista de aristas $x = \{(p, q) | (p, q) \in E'\}$ tal que $card(x) = m - 1$. Véase el ejemplo en la figura 2.

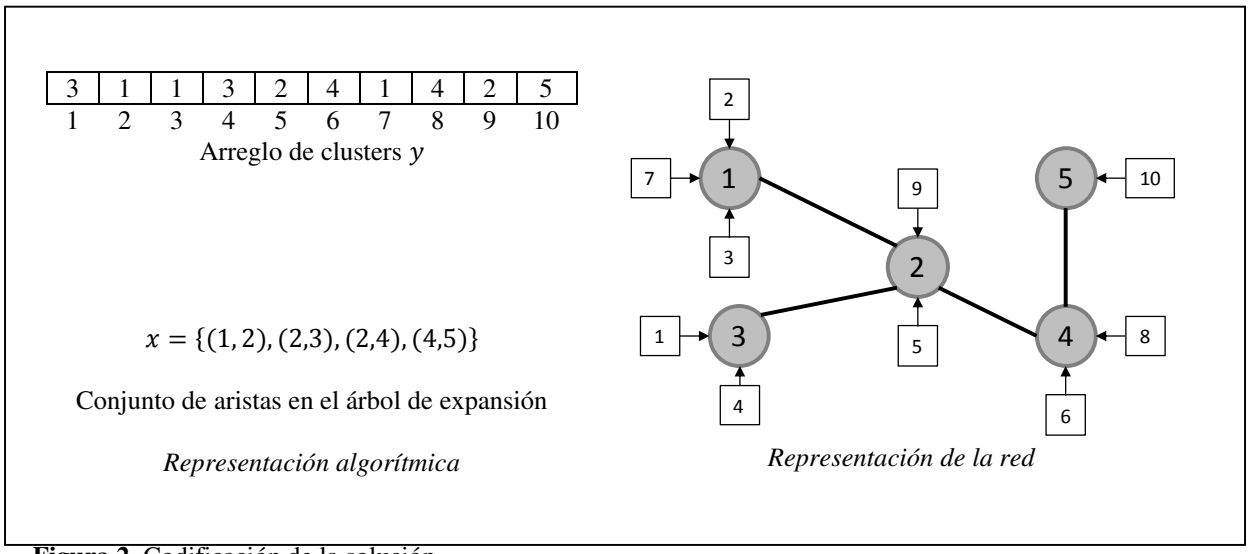


Figura 2. Codificación de la solución.

En la figura 2, se muestra una posible representación de una solución con cinco clusters y diez usuarios. Por ejemplo, $y(4) = 3$ significa que el usuario 4 deberá ser asignado al clúster 3. En la representación gráfica de la red, los círculos representan clusters mientras que los cuadrados representan los usuarios. Las líneas gruesas representan los puentes que unen dos clusters, y las flechas indican la asignación de usuarios a clusters. La solución que se muestra es factible si la restricción de la capacidad del clúster se satisface.

Desde un punto de vista algorítmico, un problema binivel se resuelve de la siguiente manera: en una iteración t el líder propone una solución y^t; restringido a esa solución, el seguidor obtiene su respuesta $x^*(y^t)$ con el objetivo de minimizar la función de nivel inferior $f_L\left(x^*(y^t)\right)$. Esto es, el seguidor reacciona racionalmente a la solución hecha por el líder y brinda su solución al líder, quien evalúa la función objetivo del nivel superior $f_U(y^t, x^*(y^t))$. Entonces, si es necesario el líder cambia su decisión para proponer una solución diferente y^{t+1}. El procedimiento se repite hasta que se cumpla un criterio de paro. La dificultad en la resolución de los problemas de programación binivel surge de la necesidad de resolver el nivel inferior en cada iteración de un algoritmo que explora soluciones del nivel superior. De ahí que el tiempo computacional consumido para resolver el nivel inferior debe ser minimizado. Por lo general, si es posible el nivel

inferior se resuelve de manera óptima; en los casos en que el problema de nivel inferior es NP-duro o un problema de combinatoria fuerte, se supone que, dado que el seguidor reacciona de forma racional a la decisión de un líder, se puede conformar con obtener una solución aceptable (de buena calidad con un bajo costo computacional). Es importante mencionar que al considerar este enfoque se alcanzará un equilibrio de Stackelberg aceptable.

Ahora, dicho procedimiento aplicado a problemas binivel se adaptó para el BLANDP de la siguiente manera: dada una nueva asignación en el arreglo y, el mínimo retraso promedio mensaje árbol de expansión $x^*(y)$ se obtiene mediante la resolución de una variante del mínimo árbol cubriente (MST, por sus siglas en inglés). Con el fin de crear soluciones factibles para el problema del mínimo retraso promedio de mensajes en árbol cubriente, se propone un algoritmo constructivo voraz. El método constructivo en cada paso va introduciendo exactamente una arista a la solución parcial actual (es decir, la adición de un clúster para el árbol actual). Antes de describir los algoritmos que debe tenerse en cuenta que en la ecuación (1), donde el tráfico a clúster k se calcula, solo el tercer término depende del árbol de expansión. Por lo tanto, se define el tráfico parcial en clúster k como $L'_k = \sum_{p \in V} t_{pk} + \sum_{q \in V \setminus \{k\}} t_{kq}$, y el tráfico parcial que fluye en cada par de puentes como $F'(x)^{(p,q)} = t_{pq} + t_{qp}$. También, el retraso promedio mensaje parcial causado por la arista $e_{(p,q)} \in E$ se define como:

$$Q\left(e_{(p,q)}\right) = \frac{1}{\Gamma}\left[\frac{L'_p}{C_p - L'_p} + \frac{L'_q}{C_q - L'_q} + F'^{(p,q)} b_{pq}\right] \forall (p,q) \in E \quad (11)$$

El algoritmo constructivo propuesto es un proceso iterativo que es similar al algoritmo de Kruskal. En la iteración k, el algoritmo selecciona una arista $e_k = e_{(p,q)}$ con el mínimo retardo promedio por mensaje $Q(e_k)$ parcial de la serie de aristas que no han sido incluidas en el árbol. Este criterio se puede utilizar para aumentar el tamaño del árbol bajo construcción manteniendo la factibilidad (es decir, que al incluir una arista no se forme un ciclo) y después se añade dicha arista al árbol actual $T = (V, E^{k-1} \cup \{e_k\})$ donde los vértices del árbol serán los $T_V = \{v \in V : v \text{ está en el árbol actual } T\}$ y las

aristas que forman parte del árbol son $T_E = \{(p,q) \in E: (p,q)$ está en el árbol actual $T\}$, teniendo en cuenta que $T_E \subseteq E'$. El algoritmo se detiene cuando $|T_E| = m - 1$. Después de que se añade una arista $e_{(p,q)}$ al árbol, el retardo del mensaje promedio actual se actualiza con la siguiente fórmula:

$$Q\left(e_{(p,q)}, T\right) = \frac{1}{\Gamma}\left[\sum_{p \in T_V} \frac{L'(x)_p}{c_p - L'(x)_p} + \sum_{(p,q) \in T_E} F'(x)^{(p,q)} b_{pq}\right] \tag{12}$$

Una vez construido el árbol, se obtiene $x^*(y)$ y ahora si ya se puede hacer la evaluación de la función del nivel inferior $f_L(x^*(y))$. Entonces, teniendo en cuenta la decisión y del líder y la reacción racional del seguidor $x^*(y)$ el costo asociado a la función objetivo del nivel superior $f_U(y, x^*(y))$ se calcula como sigue:

$$f_U\left(y, x^*(y)\right) = \sum_{(p,q) \in E'} w_{pq} + \sum_{i=1}^{n} \sum_{p \in \{V: y(i) = p\}} \alpha_{ip} y_{ip} \tag{13}$$

3.5 DESCRIPCIÓN DEL ALGORITMO PROPUESTO

A continuación, vamos a describir los componentes específicos para el GA propuesto:

Población inicial: Con el fin de generar una población diversa los individuos son creados aleatoriamente. Un individuo particular y_k es creado de la siguiente manera: Para cada de los n usuarios se genera un número aleatorio entre 1 y $|V|$, donde $|V|$ representa el número total de clusters en la red, y se agrega al individuo actual. Este proceso se repite hasta que la población inicial se completa, i.e. k alcanza el número deseado para el tamaño de la población. Es importante mencionar que si los individuos se crean en la forma descrita, la factibilidad está garantizada.

Después de que la población inicial es creada, para cada individuo se obtiene la reacción racional del seguidor $x^*(y)$; es decir, se resuelve el problema del nivel inferior. Después de esto, ahora sí, la aptitud $f_U\big(y, x^*(y)\big)$ puede ser evaluada.

Selección: Para poder realizar dicha selección de los individuos en el GA debe definirse una métrica para evaluar la aptitud. Este valor de aptitud mide la calidad de los individuos y permite compararlos entre sí. Como nosotros estamos resolviendo un problema de programación bi-nivel el valor de aptitud considerado deberá ser el valor de la función objetivo del líder, es decir, el valor dado por la fórmula (13).

Se seleccionó una estrategia de selección por torneos debido a su eficiencia para evitar la convergencia prematura del algoritmo. Un encuentro dentro de un torneo consiste en seleccionar un individuo y compararlo aleatoriamente con otro individuo de la población, entonces se comparan sus respectivos valores de aptitud y se identifica el ganador. El ganador será el individuo con el mejor valor de aptitud, esto es, el individuo con menor costo de conexión. Estos encuentros se realizan para todos los individuos en la población actual. Con el fin de permitir que los individuos con el mejor valor de aptitud permanezcan en la población, se llevarán a cabo un número predefinido de torneos. Es digno de notar que si se hacen muy pocos torneos, entonces el ranking de los individuos tiende a tener mucha aleatoriedad. Por otra parte, si se llevan a cabo muchos torneos, el ranking se hará con preferencia a los mejores individuos eliminando la diversidad requerida para los GA. Después, los individuos de la población con mejor aptitud será seleccionada para entrar a los operadores genéticos. En otras palabras, se hace una selección elitista para tratar de mejorar la población en los operadores genéticos.

El algoritmo considera dos operadores genéticos: cruzamiento y mutación (ver figura 3). Por lo tanto, para cada uno de los individuos elegidos en la fase de selección un número aleatorio entre 0 y 1 es generado. Si el número aleatorio es menor o igual que

un parámetro previamente definido, entonces el individuo entrará al operador de cruzamiento; de lo contrario, entrará a la mutación.

Cruzamiento: Este es el principal operador genético, es por esto que la probabilidad de entrar en esta fase debe ser mayor que 0.50. El cruzamiento simula la reproducción entre dos individuos, llamados padres. El procedimiento es como sigue: el individuo actual se apareja aleatoriamente con otro individuo de la población (donde por población nos referimos a la población completa, no solo la mitad elitista correspondiente con los individuos seleccionados). Entonces, ambos padres se combinan con el fin de producir dos hijos. Se implementó hacerlo mediante un punto de cruzamiento estándar; dicho punto es seleccionado aleatoriamente por el primer padre (P1) y también considerado por el segundo padre (P2). Uno de los hijos heredará la primera parte de P1 y la segunda parte de P2; el otro hijo será creado en el sentido contrario.

Mutación: En el caso cuando un individuo ha entrado en esta fase, deberá causarle un pequeño cambio en su codificación. Este cambio aleatorio gradualmente incorporará nuevas características a la población, lo cual permite explorar nuevas regiones del espacio solución. Recuerde que en el cruzamiento produce dos hijos con las mismas características que los padres. Sin embargo, la mutación toma un lugar importante en el algoritmo ya que permite obtener mayor diversidad entre los individuos. La mutación se realiza mediante la selección de un componente de la solución actual y después se realiza un cambio aleatorio en ese componente seleccionando otro número entre 1 y $|V|$; esto es, un usuario específico es conectado a otro clúster.

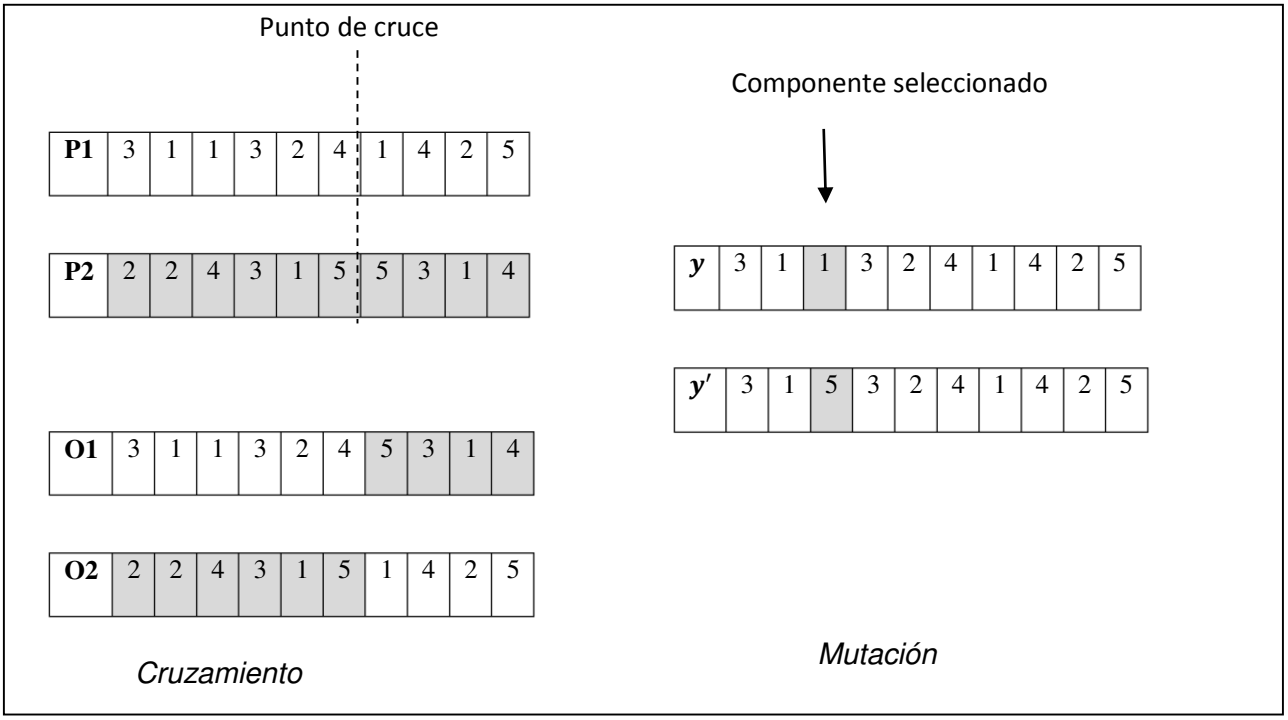


Figura 3. Operadores Genéticos

CAPÍTULO 4

EXPERIMENTACIÓN COMPUTACIONAL

4.1 DESCRIPCIÓN DE LOS EXPERIMENTOS

Las pruebas computacionales pueden ser divididas en tres partes principales. Primero, usamos como base un conjunto de tres instancias reportadas en Kim *et al.* (2009). En este conjunto de instancias los usuarios en la red varían desde 8 hasta 50, y los clusters varían desde 4 hasta 10. Después, se presenta una discusión sobre la calidad de la solución alcanzada -en el mismo conjunto de instancias- por el algoritmo Nash-Genético existente en la literatura. Finalmente, se creó un nuevo conjunto de 10 instancias de mayor tamaño para validar la robustez del algoritmo propuesto. El nuevo conjunto de instancias varía su tamaño desde 60 hasta 300 usuarios y desde 15 hasta 50 clusters en la red.

Se realizó una afinación de los parámetros involucrados en el algoritmo propuesto para resolver las instancias existentes y las instancias generadas. Dicho estudio se muestra más adelante en la sección correspondiente. Todas las instancias consideradas en nuestra experimentación están disponibles si alguien las solicita para comparar algún otro algoritmo. Ambos conjuntos de instancias se utilizaron para analizar el desempeño de la metodología de solución en este trabajo.

Por último, vamos a describir el ambiente computacional utilizado. Toda la experimentación realizada en este trabajo se implementó en C++ utilizando el entorno de programación Microsoft Visual Studio 2010 a través de un sistema operativo Windows 7. Todos los experimentos se llevaron a cabo en una computadora HP Compaq 6000 Pro PC con un procesador Pentium Dual-Core a 3,00 GHz y 2,00 GB de RAM.

4.2 EXPERIMENTACIÓN CON LAS INSTANCIAS CONOCIDAS

Como se ha mencionado anteriormente, en la literatura sólo existen tres instancias para este problema. Sin embargo, las instancias están incompletas ya que algunos parámetros solamente vienen descritos sin presentar los datos exactos, esto es, se indica una distribución de probabilidad para los parámetros pero no se especifica su valor. Por lo tanto, la información que falta se ha generado con el mismo procedimiento descrito en Kim *et al.* (2007) y Kim *et al.* (2009). Los valores de los parámetros de los problemas considerados se especifican en la Tabla 4.1.

Tabla 4.1. Datos del problema

	Problema 1	Problema 2	Problema 3
Número de usuarios	8	30	50
Numero de clusters	4	6	10
Costo de conexión de clusters (w_{pq})	$w_{pq} \sim U(100,250)$		
Costo de conexión de usuarios(α_{ip})	$\alpha_{ip} \sim U(1,100)$		
Capacidad (C_p)	50	300	500
Tiempo de respuestas de clusters (b_{pq})	0.1	0.1	0.1

Antes de llevar a cabo la experimentación computacional, se realizó la calibración de los parámetros que intervienen en el algoritmo genético propuesto. Los principales parámetros son: el número de generaciones (G), el tamaño de la población (P), la probabilidad (π) de entrar a la fase de cruzamiento o mutación y el número de torneos realizados en la fase de selección. Este último fue fijado en 5 debido a que es un valor comúnmente recomendable para tratar de mantener un equilibrio entre la diversificación e intensificación de las soluciones. Para los otros tres parámetros, i.e. G, P and π, se usaron algunas pruebas numéricas preliminares para establecer los valores de los parámetros requeridos. Las pruebas preliminares consisten en realizar corridas con diferentes valores de los parámetros y después, llevar a cabo algunas comparaciones numéricas de los resultados obtenidos.

En primer lugar, en función del tamaño de las instancias nosotros establecemos cuatro diferentes posibles valores para P, estos son 100, 150, 200 and 300. Entonces, hacemos lo mismo para el parámetro π seleccionando 0.50, 0.60, 0.75 y 0.90. El valor de G se estableció como 500 para las pruebas preliminares y después se reducirá hasta donde sea necesario. Como un ejemplo, las gráficas que muestran el comportamiento de los resultados numéricos obtenidos para los tres problemas se ilustran en las Figuras 4.1-4.3.

Esta parte del trabajo experimental fue llevado acabo para analizar el comportamiento de cada combinación de parámetros propuesta para las tres instancias consideradas. Debido a la aleatoriedad presente en la metodología, para cada una de las diferentes combinaciones de parámetros el algoritmo genético fue corrido diez veces.

Con el fin de seleccionar los parámetros para el algoritmo genético, se realizó un diseño factorial completo. El diseño está compuesto por 3 tratamientos y 4 niveles con 10 réplicas. Se consideraron dos variables de respuesta, la función objetivo del líder y el tiempo requerido. Los resultados del diseño experimental mostraron que los tres factores tienen un efecto significativo en ambas variables de respuesta. Cabe mencionar que se consideró un máximo de tiempo necesario para resolver el problema con el fin de descartar algunas combinaciones de niveles de los tratamientos.

Por otro lado, los resultados obtenidos de las pruebas de cálculo descritos en el diseño factorial completo se analizaron gráficamente. En las Figuras 4.1-4.3 está graficado el promedio de los valores obtenidos después de diez corridas de cada una de las combinaciones para los Problemas 1-3, respectivamente. Los ejes corresponden al número de generaciones (eje de las abscisas) y al valor de la función objetivo del líder (eje de las ordenadas). Los resultados obtenidos de variar la probabilidad en los operadores genéticos (π) se pueden observar en cada una de las gráficas. También, cada gráfica corresponde a uno de los diferentes tamaños de la población (P).

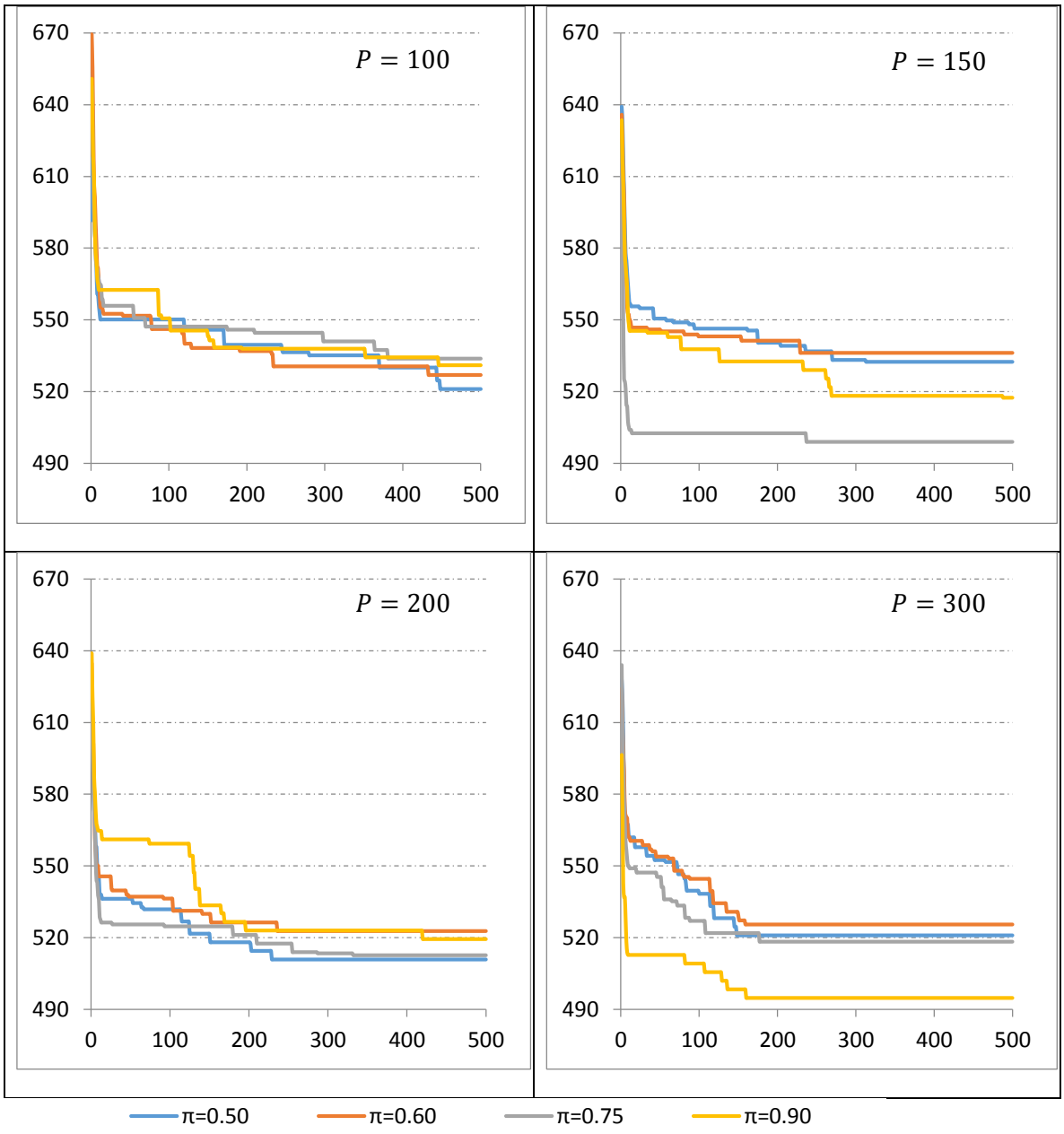


Figura 4.1. Afinación de parámetros para el Problema 1.

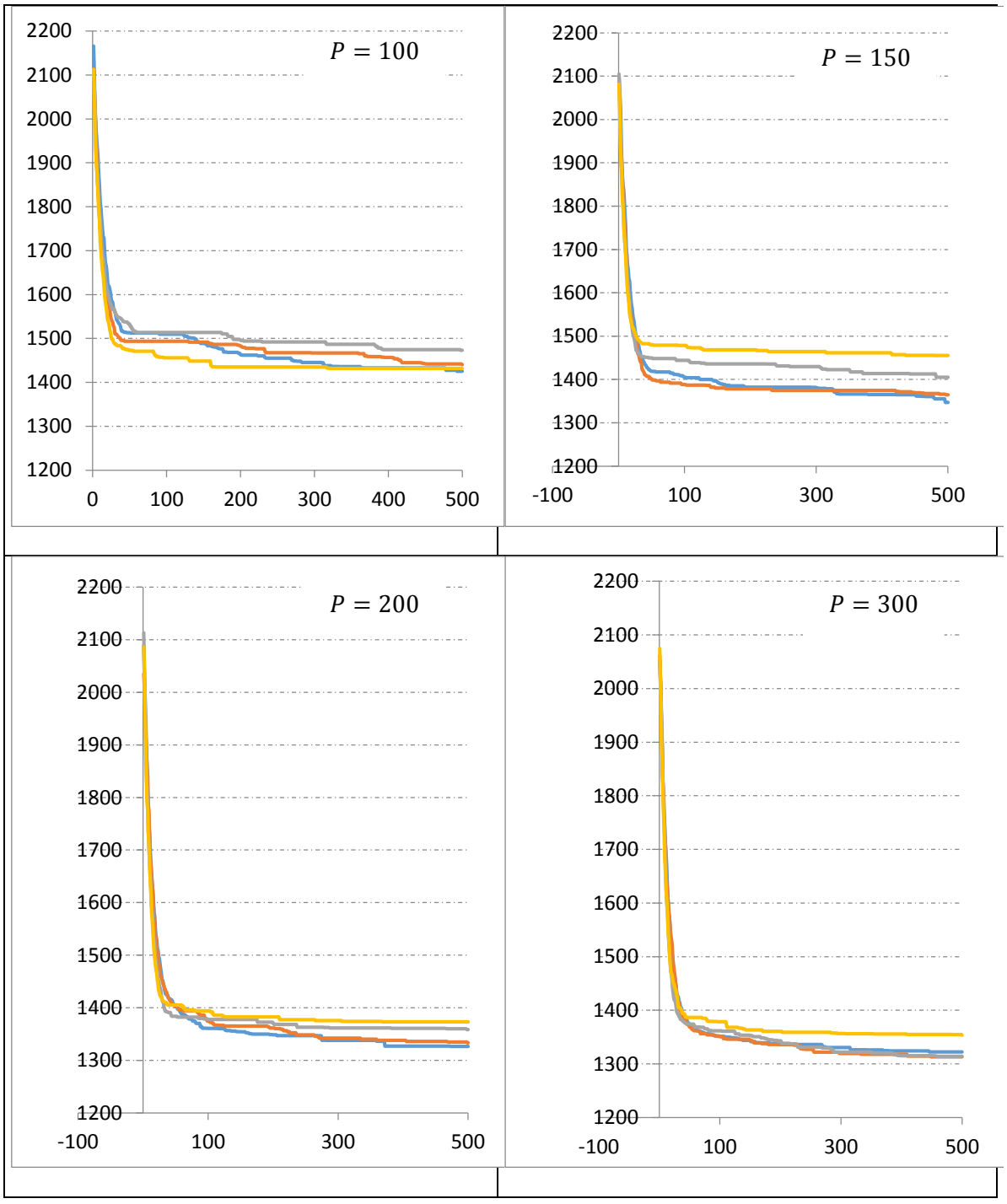


Figura 4.2. Afinación de parámetros para el Problema 2.

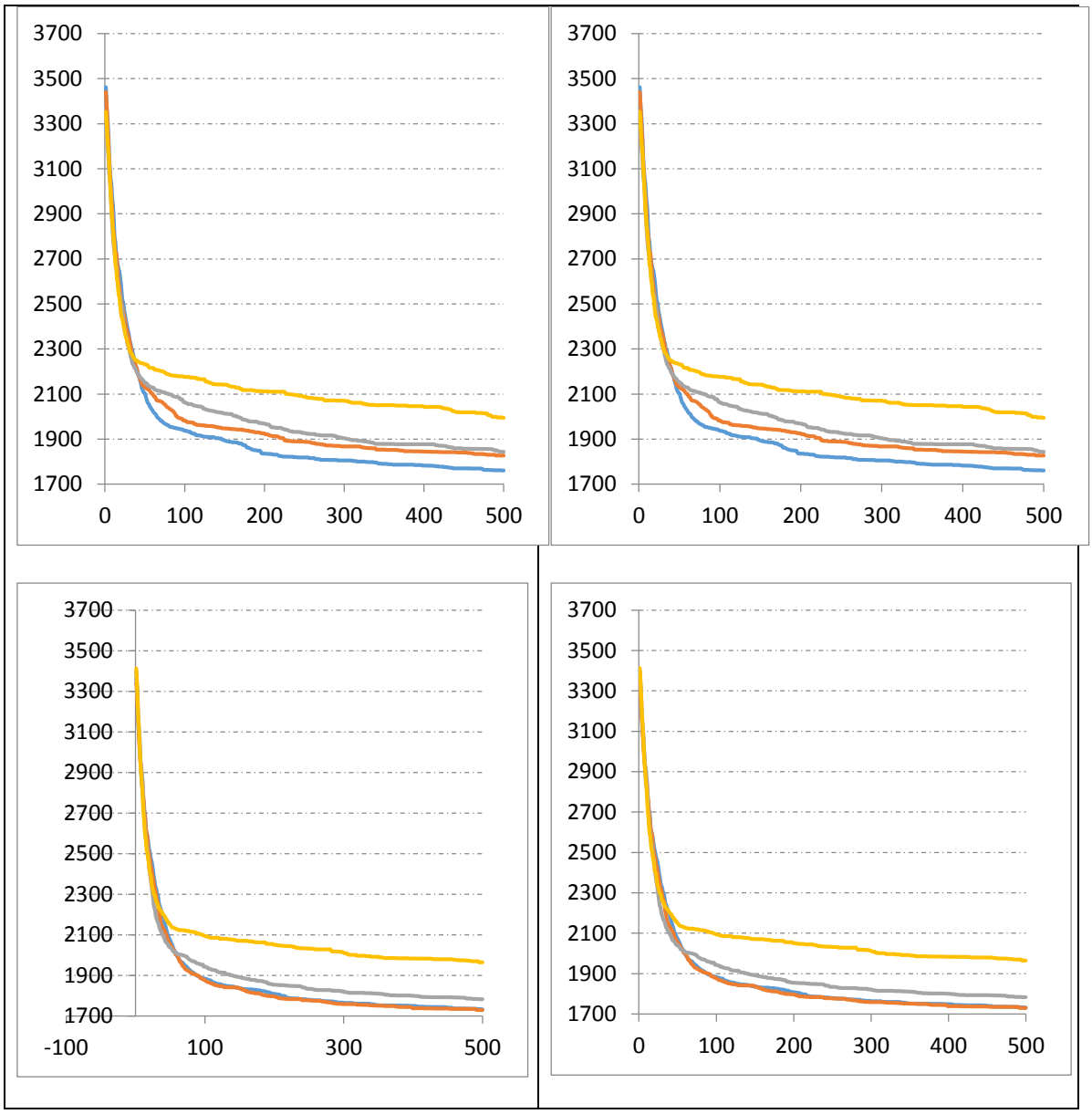


Figura 4.3. Afinación de parámetros para el Problema 3.

Cuando comparamos las diferentes combinaciones de parámetros a partir de los resultados que se muestran en las Figuras 4.1-4.3, se puede observar que su eficiencia es bastante similar en cuanto a la calidad de la solución y al tiempo de solución que se refiere (es decir, que el tiempo está directamente relacionado con el número de generaciones). Parece ser complicado identificar algunas combinaciones de parámetros que claramente dominan a las otras.

Por lo tanto, considerando que calcular la reacción racional del seguidor no es sencillo, más bien es difícil debido a la complejidad inmersa en el problema, una se procura tener una población pequeña. Por otro lado, para el número de generaciones parece que con un valor intermedio se obtienen buenos resultados porque las corridas grandes incurrirán en un alto tiempo computacional. También, se pueden identificar algunos puntos críticos donde la calidad de la solución no mejoraría más.

Después de haber analizado los resultados obtenidos del diseño de experimentos y respaldado por la ilustración gráfica mostrada en las Figura 4.1, el ajuste de parámetros realizado para el Problema 1 es $\pi = 0.75$, $P = 150$ and $G = 300$. Se puede observar que, mientras más alta sea la probabilidad de π, el algoritmo alcanza mejores valores de la función objetivo del líder. Teniendo en cuenta que la misma solución del líder puede provocar diferentes reacciones racionales del seguidor, la probabilidad de entrar al cruzamiento o a la mutación es $\pi = 0.75$. Además, como se mencionó arriba, debido a que el tamaño de la población P afecta negativamente al tiempo requerido, entonces un valor pequeño de P es preferido, i.e. $P = 150$.

Finalmente, el número de generaciones también tiene un impacto en el tiempo requerido, entonces 300 generaciones parece ser un valor eficiente basado en el tiempo de cálculo y en el valor de la función objetivo del líder. Por ejemplo, el tiempo promedio consumido para las 500 generaciones en las diez corridas de cada generación fue 4.5, 6.7, 9 y 13.7 segundos para 100, 150, 200 y 300 individuos en la población,

respectivamente. El mismo análisis fue hecho para los otros dos problemas considerados. El resultado del ajuste de parámetros se muestra en la Tabla 4.2.

Tabla 4.2. Ajuste de parámetros.

	Problema 1	Problema 2	Problema 3
Probabilidad de entrar a los operadores genéticos (π)	0.75	0.50	0.60
Tamaño de la población(P)	150	200	200
Numero de generaciones (G)	300	400	500

Después de haber calibrado los parámetros para cada una de las instancias existentes, se realizaron 50 corridas del código con el fin de evaluar la calidad del algoritmo genético propuesto. En la Tabla 4.3 se muestran los resultados de la experimentación computacional. La columna "Mejor" representa el mejor valor obtenido a partir de las 50 corridas para cada problema. En la columna "Promedio", se muestra el promedio de la calidad de las soluciones obtenidas en las 50 corridas y en la columna "Peor" se indica el mayor costo obtenido. A continuación, la columna "Holgura" se calcula como $Holgura = \frac{|Best - Average|}{Best} \times 100\%$. La desviación estándar de la muestra se presenta en la columna "Desv. Est". Las columnas "# Mejor" y "% Mejor" indican el número total de veces y el porcentaje de veces en que se alcanzó el mejor valor, respectivamente. Por último, la columna "Tiempo" indica el promedio de tiempo (en segundos) para resolver 50 veces cada problema.

Tabla 4.3. Resultados numéricos para el cálculo

	Mejor	**Promedio**	**Peor**	**Holgura**	**Desv. Est**	**# Mejor**	**% Mejor**	**Tiempo**
Problema1	493	498.94	502	1.20	4.31	27	54	4.230
Problema 2	1203	1226.30	1282	1.94	18.04	22	44	13.951
Problema 3	1602	1652.89	1738	3.18	38.6	18	36	54.877

En la Tabla 4 se puede observar que para el Problema 1, el algoritmo alcanza el máximo valor en más de la mitad de las 50 corridas. Por otra parte, el promedio de todas las corridas está muy cerca del mejor valor obtenido y la desviación estándar indica que los valores están muy cercanos de la media; la pequeña holgura obtenida (1,20%) confirma el buen desempeño del algoritmo desarrollado en este problema. El tiempo promedio que se consume es 4.23 segundos.

Los resultados para el Problema 2 indican que a pesar del aumento esperado en el tiempo computacional (casi 14 segundos), el algoritmo alcanza una holgura muy aceptable entre el mejor valor obtenido y el promedio de las 50 corridas. Esta holgura es inferior al 2%. Además, se obtuvo el mejor valor en casi la mitad de la experimentación (en 22 de las 50 corridas).

Por último, la experimentación numérica realizada para el Problema 3 no era tan buena como las anteriores, pero los resultados siguen siendo razonables. El mejor valor se alcanzó en 18 de las 50 corridas, mientras que la holgura se incrementó a 3.18% y el tiempo consumido fue de 54.9 segundos. Estos resultados fueron claramente afectados por la dificultad de encontrar reacción racional del seguidor.

4.3 COMPARACION ENTRE LOS ALGORITMOS STACKELBERG-GENÉTICOS Y LOS NASH-GENÉTICOS

En esta subsección, se hace una discusión sobre la calidad y factibilidad de las soluciones obtenidas por el algoritmo Stackelberg-Genético desarrollado en este trabajo (SG, de aquí en adelante) y por el algoritmo Nash-Genético (NG, de aquí en adelante) propuesto en Kim *et al.* (2009).

Las soluciones consideradas por el SG son las que se presentan en subsección 4.2. Por otra parte, para la obtención de las soluciones del NG nosotros emulamos el algoritmo

descrito en Kim *et al.* (2009) con el fin de resolver el BLANDP. A continuación, se muestra una breve descripción general del algoritmo NG. Sea $(y \mid x)$ el par ordenado que representa una posible solución para un problema de optimización bi-objetivo. Entonces y denota el subconjunto de variables controladas por el líder y optimizadas de acuerdo a los costos de conexión. Del mismo modo x denota el subconjunto de variables controladas por el seguidor y optimizadas con respecto al promedio de retardo en la red. Según la perspectiva Nash, el líder optimiza $(y \mid x)$ con respecto a su función objetivo mediante la modificación y, mientras que x es fijada por el seguidor, es decir, el líder encontrará $y^*(x)$. Simétricamente, el seguidor optimiza $(y \mid x)$ con respecto a su función objetivo seleccionando x mientras que y es fijada por el seguidor, es decir, el seguidor encontrará $x^*(y)$. Es importante notar que a pesar de que estamos refiriéndonos a un líder y un seguidor, bajo el enfoque de Nash, ambos están en el mismo nivel de decisión, es decir, no hay jerarquía entre ellos. Luego, de la misma manera que en Kim *et al.* (2009), dos poblaciones diferentes son consideradas. La primera de ellas se denomina $pop1$ y corresponde a las asignaciones asociadas con la conexión de los usuarios a los clusters; mientras que la segunda se llama $pop2$ y corresponde a los árboles cubrientes x que resultan de la conexión de los clusters. En cada población la aptitud del individuo se evalúa con respecto a la función objetivo correspondiente, es decir, el objetivo del líder o el del seguidor.

Sea y_{k-1} el mejor valor encontrado por el líder en la generación $k-1$ y x_{k-1} es el mejor valor encontrado por el seguidor en la generación $k-1$. En la generación k, el líder optimiza y_k considerando como fija a x_{k-1} para poder evaluar $(y \mid x)$. De la misma manera, el seguidor optimiza x_k considerando fijo el valor de y_{k-1} con el fin de evaluar $(y \mid x)$. Después del proceso de optimización, el líder envía el mejor valor y_k al seguidor que lo utilizará en la generación $k+1$; a continuación, el seguidor hace el mismo procedimiento con respecto al líder. El equilibrio de Nash se alcanza cuando ni el líder ni el seguidor pueden mejorar sus propios criterios sin afectar a los intereses de otro. Este procedimiento se ilustra en la Figura 4.4.

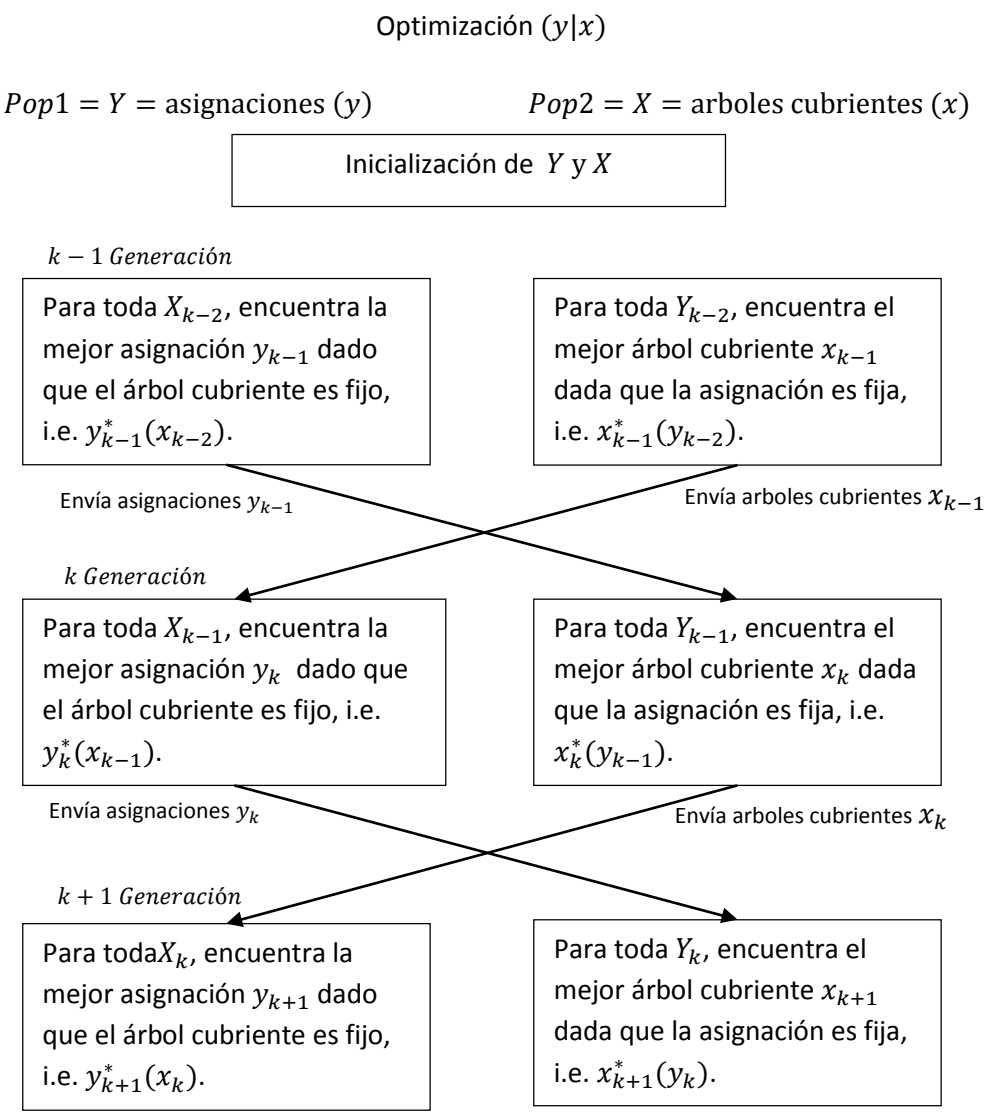


Figura 4.4. Proceso del algoritmo Nash-Genético.

Es importante mencionar que para el algoritmo Nash-genético que se implementó los operadores genéticos (cruzamiento y mutación) y la fase de selección son los mismos que los descritos en el capítulo 3 de esta tesis.

Con la finalidad de mostrar el desempeño del algoritmo NG, hemos resuelto los mismos tres problemas existentes y comparamos los resultados obtenidos. Cada instancia se ejecutó 50 veces, como en el algoritmo SG. Sea $(pop1, pop2)$ quien denote el tamaño

seleccionado para las poblaciones del líder y seguidor, respectivamente. Para el Problema 1, las poblaciones son (10,10) debido al hecho de que solamente existen 16 árboles cubrientes. Para los Problemas 2 y 3 el tamaño de las poblaciones es de (100,100). El algoritmo se detiene cuando el mejor individuo para ambas poblaciones es el mismo, esto es, cuando no se realiza alguna mejora en al menos una función objetivo. Los resultados para los algoritmos SG y NG se presentan en la Tabla 4.4. La descripción de las columnas es la misma que la de la Tabla 4.3.

Tabla 4.4. Resultados numéricos de los algoritmos SG y NG.

	SG				**NG**			
	Mejor	**Promedio**	**Holgura**	**Tiempo**	**Mejor**	**Promedio**	**Holgura**	**Tiempo**
Problema1	493	498.94	1.20	4.230	635	635.00	0.00	0.751
Problema2	1203	1226.30	1.94	13.951	1217	1234.74	1.44	27.655
Problema3	1602	1652.89	3.18	54.877	1518	1716.82	11.58	170.469

Los principales comentarios acerca de los valores que se muestran en la Tabla 4.4 tienen que ver con el tiempo computacional y con el valor de la función objetivo del líder. Cuando la instancia contiene a unos clusters que exigen un proceso evolutivo en la población del seguidor, el tiempo requerido para el NG se incrementa. Esto es causado claramente por la existencia de dos poblaciones, que requieren que entren a los operadores genéticos para evolucionar. En cambio, si el tamaño de la instancia es pequeño, no hay necesidad de evolucionar tanto en la población del seguidor.

Además, no hay evidencia para indicar si aumenta o disminuye valor de la función objetivo del líder. Sin embargo, es muy importante tener en cuenta que se podría encontrar soluciones eficientes -en términos de Pareto- para valores mejores de la función objetivo, pero en la mayoría de las veces estas soluciones no van a estar en la región inducible del problema binivel. Como se ha mencionado en el apartado de la literatura relacionada, la solución óptima binivel no se encuentra necesariamente en el

conjunto de soluciones eficientes de un problema bi-objetivo. Entonces, no podemos hacer una comparación válida acerca del valor de la función objetivo alcanzado por el algoritmo SG y el algoritmo NG ya que, en general, las soluciones no serán factibles del problema binivel.

La cuestión importante aquí es mostrar la diferencia significativa que existe al resolver un problema de programación binivel sin tomar en cuenta de manera adecuada la jerarquía entre los roles del líder y seguidor. Además, el principal detalle en considerar el enfoque del NG es que la obtención de $y^*(x)$ en la población del seguidor puede no ser adecuado para resolver los problemas binivel, en los cuales se debe encontrar la mejor asignación para un árbol cubriente en particular y esto no concuerda con la jerarquía predefinida considerada en el problema de programación binivel. Entonces, los algoritmos Nash-Genéticos parecen ser mejor alternativa para resolver los problemas de programación multi-objetivo.

4.4 ROBUSTEZ DEL ALGORITMO STACKELBERG-GENÉTICO

El objetivo de esta sección es mostrar que el rendimiento del algoritmo es estable y eficiente. Para conseguir esto, se generó un nuevo conjunto de 10 instancias de mayor tamaño de manera aleatoria mantenimiento la misma estructura que las instancias existentes. Es decir, seguimos considerando el costo de conexión de usuarios y clusters como $\alpha_{ip} \sim U(1,100)$ y $w_{pq} \sim U(100,250)$, respectivamente. El tiempo de respuesta del clúster se estandariza como 0.1 para todas las instancias. El resto de los datos para cada instancia están dados en la Tabla 4.5.

Tabla 4.5. Datos para las instancias generadas.

Instancia	Usuarios	Clusters	Capacidad
G1-1	60	15	600
G1-2	70	20	750
G1-3	80	20	1000
G1-4	90	25	1000
G1-5	100	25	1500
G1-6	150	30	2500
G1-7	200	30	4000
G1-8	250	40	5000
G1-9	300	20	6500
G1-10	100	50	1200

Ya que el segundo conjunto de instancias contiene problemas de tamaño más grande que los anteriores, se consideró un mayor número de posibles valores para cada parámetro. Esto es, para P consideramos 100, 200, 300, 400 y 500. Similarmente para π se probaron 0.50, 0.60, 0.70, 0.80 y 0.90. Los valores para el número de generaciones G se establecieron como 500, 1000, 1500 y 2000, con el fin de hallar el valor más apropiado para cada instancia. Se llevaron a cabo pruebas preliminares de la misma manera que para las instancias anteriores. Luego, se llevó a cabo un diseño factorial completo al igual que en el caso anterior. Los resultados fueron graficados de la misma manera como se muestra en las Figuras 4.1-4.3. Entonces, el ajuste de parámetros apropiado para cada instancia se presenta en la Tabla 4.6.

Tabla 4.6. Ajuste de parámetros para las instancias generadas.

Instancia	π	P	G
G1-1	0.70	100	500
G1-2	0.70	200	500
G1-3	0.80	200	500
G1-4	0.70	300	1000
G1-5	0.60	300	1000
G1-6	0.60	400	1000
G1-7	0.50	400	1500
G1-8	0.50	500	1500
G1-9	0.60	300	1000
G1-10	0.50	500	1500

Además, en la Tabla 4.7 se muestran los resultados numéricos de la experimentación computacional considerando los parámetros descritos en la Tabla 4.6. Los encabezados de la Tabla 4.7 son análogos a los de la Tabla 4.3. De la misma manera que para el conjunto de instancias existentes, se llevaron a cabo 50 corridas para el nuevo conjunto de instancias.

Tabla 8. Resultados numéricos para las nuevas instancias generadas.

Instancia	Mejor	Promedio	Peor	Holgura	Desv. Est	#Mejor	%Mejor	Tiempo
GI-1	2773	2927.62	3272	5.58	165.12	14	28	66.711
GI-2	3838	3940.85	4457	2.68	175.38	20	40	82.532
GI-3	3745	4018.69	4645	7.31	321.93	12	24	85.726
GI-4	4728	4972.73	5589	5.17	303.82	16	32	105.248
GI-5	5397	5662.83	6227	4.92	307.00	10	20	111.392
GI-6	7369	7891.67	9167	7.09	693.39	21	42	163.966
GI-7	9529	10353.87	11534	8.65	711.07	9	18	206.473
GI-8	14669	15189.88	16025	3.55	533.99	6	12	289.007
GI-9	12902	13236.83	14726	2.60	470.31	16	32	151.057
GI-10	10043	10484.02	11376	4.39	332.24	3	6	178.264

De la Tabla 4.7 podemos apreciar que el algoritmo Stackelberg-Genético tiene un desempeño estable. La holgura entre el mejor valor alcanzado de la función objetivo del líder y el promedio para las 50 corridas es menor que el 9% para todas las instancias. La desviación estándar es pequeña en 8 de las 10 instancias; esto indica que en la mayoría de las corridas el algoritmo converge a una región que contiene soluciones de alta calidad. El porcentaje de veces que el algoritmo repite la mejor solución obtenida es aceptable, de 18% a 42% de las corridas. Cuando el número de clusters aumenta, como, en GI-8 y G10, el algoritmo disminuye su rendimiento alcanzando el mejor valor solo en 6 y 3 corridas, respectivamente. Este comportamiento se debe al aumento

significativo en el espacio de decisión del seguidor. Además de que el método para obtener soluciones del nivel inferior es heurístico, entonces esto provoca que haya una gran variabilidad. Se esperaba un aumento en el tiempo requerido, ya que el tamaño de las instancias generadas aumenta. Sin embargo el tiempo requerido parece tener un incremento polinomial. Aunque esto es principalmente afectado por el número de generaciones y en menor grado por el tamaño de la población.

Finalmente, vale la pena remarcar que un incremento en el número de clusters aumenta exponencialmente el número de árboles posibles, es decir, el espacio de decisión del seguidor. La fórmula de Cayley puede ser aplicada para encontrar el número de árboles factibles asociados con la decisión del seguidor. Por lo tanto, el rendimiento del algoritmo es afectado negativamente por este hecho. La heurística considerada para encontrar la reacción racional del seguidor es un tema principal para futuras investigaciones.

CAPÍTULO 5
CONCLUSIONES Y FUTURAS INVESTIGACIONES

5.1 CONCLUSIONES

En esta tesis, se propuso un modelo de programación binivel para analizar un problema de diseño topológico de una red de área local. En este problema, el líder decide la ubicación de usuarios a clusters con el fin de minimizar el costo de conexión; mientras el seguidor conecta los clusters formando un árbol de expansión que minimice el retardo promedio en la red. Para resolver este problema de manera eficiente, se diseñó un algoritmo genético que considera un equilibrio aceptable de Stackelberg. Este algoritmo considera que el problema del nivel inferior no puede ser resuelto de manera exacta en todos los casos, por lo tanto la reacción racional del seguidor necesita ser definida de una manera alternativa. En el sentido estricto, las soluciones factibles binivel deben tener la respuesta óptima del seguidor para la decisión correspondiente del líder; como ya se mencionó en este caso no es posible, así que se asume que una solución aceptable en el nivel inferior es suficiente. Para lidiar con esto último, implementamos un procedimiento heurístico que parece ser eficiente en términos de la calidad de la solución y el tiempo requerido para obtenerla.

Las pruebas computacionales se llevaron a cabo tomando como base unas instancias existentes en la literatura para este problema y un nuevo conjunto de instancias generadas aleatoriamente. Las conclusiones que podemos hacer después de analizar los resultados son que el mejor valor de la función objetivo del líder se encontró varias veces y el tiempo computacional consumido es muy aceptable para un problema de esta naturaleza. La robustez del algoritmo propuesto se mostró mediante experimentación numérica realizada a instancias de tamaño diferente. El rendimiento del algoritmo es

estable sin tener mucha variación debido a los componentes de las diferentes instancias.

Es importante mencionar que, dado que no existe algún método eficiente para optimizar problemas binivel de diseño de red a gran escala, un enfoque iterativo de optimización para asignación se ha utilizado en los problemas de diseño de red (por ejemplo, en el ajuste de la señal de tráfico e incremento de capacidades en las conexiones). Este algoritmo consiste en iterar entre el problema de optimización de nivel superior fijando los valores de las variables de decisión de nivel inferior, y luego considerar el problema de optimización del nivel inferior (tiempo medio de retraso por el tráfico) con valores fijos para la variable de decisión de nivel superior. Sin embargo, se demostró teóricamente y empíricamente que este algoritmo iterativo no necesariamente converge a las soluciones exactas de juegos Stackelberg, sino es más bien un algoritmo exacto y eficiente para resolver juegos Cournot-Nash, en el que cada jugador intenta maximizar su utilidad o ganancia de forma no cooperativa asumiendo que sus acciones no tendrán ningún efecto sobre las acciones de los otros jugadores (ver Fisk, 1984 y Friesz and Harker, 1985).

También, se debe mencionar que el algoritmo iterativo de optimización de la asignación presentado en Kim, *et al.* (2009), obviamente no resuelve el problema considerado en esta tesis debido a que la solución óptima del problema de nivel superior es el principal objetivo si los valores de las variables de decisión de nivel inferior están fijas. Podemos apreciar este hecho al observar la discusión presentada en la sección 4.2.

5.2 TRABAJO FUTURO

Como un área de oportunidad, hemos identificado que, dado que la dificultad inmersa al lidiar con un problema del seguidor que sea difícil y combinatorio, se tiene que proponer alguna metodología alternativa para resolverlo apropiadamente. En esta tesis, se implementó un método heurístico para encontrar la reacción racional del seguidor. Esta metodología nos podría dar diferentes respuestas del seguidor para la

solución del mismo líder, pero no necesariamente el mejor árbol de expansión. Este problema afecta a la eficiencia del algoritmo genético en la experimentación asociado con los casos de mayor tamaño. Por lo tanto, una metodología que considere un conjunto de árboles de expansión y la evaluación de los mismos con el fin de obtener la mejor respuesta del seguidor para cada decisión del líder parece ser una buena opción.

Es evidente que esta metodología podría ser muy costosa en términos de tiempo de cálculo, por lo que el uso de computación en paralelo es necesario. Por otra parte, esta alternativa puede conducirnos a diseñar un algoritmo co-evolutivo donde ambas poblaciones mejoren de forma independiente pero siempre teniendo en cuenta la jerarquía existente, es decir, para cada decisión del líder encontrar la mejor respuesta del seguidor en la población correspondiente conforme vaya evolucionando.

CAPÍTULO 6
REFERENCIAS

[1] S. Alumur, and B. Y. Kara, "Network hub location problems: The state of the art," *European Journal of Operational Research,* vol.190, no.1, pp. 1–21, 2008.

[2] K. Watcharasitthiiwat, and P. Wardkein, "Reliability optimization of topology communication network design using an improved ant colony optimization," *Computers & Electrical Engineering*, vol.35, no.5, pp. 730–747, 2009.

[3] C. Ersoy, and S.S. Panwar, "Topological Design of Interconnected LAN/MAN Networks," *IEEE Journal on Selected Areas in Communications,* vol.11, no.8, pp. 1172-1182, 1993.

[4] R. Estepa, and A. Estepa, T. Cupertino, "A productivity-oriented methodology for local area network design in industrial environments," *Computer Networks,* vol. 55, no. 9, pp. 2303–2314, 2011.

[5] B. Gavish, "Topological design of telecommunication networks-local access design methods," *Annals of Operations Research,* vol.33, no.1, pp. 17–71, 1991.

[6] F. Rothlauf, "On optimal solutions for the optimal communication spanning tree problem," *Operations Research,* vol. 57, pp. 413–425, 2009.

[7] G.R. Mateus, A.A.F. Loureiro, and R.C. Rodrigues, "Optimal network design for wireless local area network," *Annals of Operations Research,* vol.106, no.1–4, pp. 331–345, 2001.

[8] J.F. Glover, and M. Lee, "Least-cost network topology design for a new service: an application of a tabu search," *Annals of Operations Research,* vol.33, pp. 351–362, 1991.

[9] R.H. Jan, F.J. Hwang, and S.T. Cheng, "Topology optimization of a communication network subject to a reliability constraint," *IEEE Transactions on Reliability*, vol.42, no.1, pp. 63–70, 1993.

[10] H.C. Chao-Hsien Chu, G. Premkumar, "Digital data networks design using genetic algorithms". *European Journal of Operational Research,* vol. 127, no. 1, pp. 140–158, 2000.

[11] S.A. Khan, and A.P. Engelbrecht, "A fuzzy particle swarm optimization algorithm for computer communication network topology design," *Applied Intelligence,* vol.36, no.1, pp. 161-177, 2012.

[12] D. Clark, K.T. Pogran, and D.P. Reed, "An introduction to local area networks," in *Proceedings of the IEEE*, vol.68, no.6, pp. 1497 –1517, 1978.

[13] R. Elbaum, and M. Sidi, "Topological Design of Local-area Networks using Genetic Algorithms," *IEEE/ACM Transactions on Networking,* vol.4, no.5, pp. 766-778, 1996.

[14] J.R. Kim, J.U. Lee, and J.-B. Jo, "Hierarchical spanning tree network design with Nash genetic algorithm," *Computers & Industrial Engineering,* vol. 56, no.3, pp. 1040–1052, 2009.

[15] B. Kara, and V. Verter, "Designing a road network for hazardous materials transportation," *Transportation Science,* vol. 44, pp. 1595-1607, 2004.

[16] W. Jing, Z. Jianming, Jun, Huang, and Z. Min, "Multi-level emergency resources location and allocation," in *Proceedings of the 2010 IEEE International Conference on Emergency and Management Sciences (ICEMMS)*, pp.202-204, 2010.

[17] Ch.M. Feng, and Ch.Ch. Wen, "A Bi-level programming model for allocating private and emergency vehicle flow in seismic disaster areas," in *Proceedings of the Eastern Asia Society for Transportations Studies*, vol.5, pp. 1408-1423, 2005.

[18] L. LeBlanc, "Mathematical programming algorithms for large scale network equilibrium and network design problems," Ph.D. Dissertation. Department of Industrial Engineering and Management Sciences, Northwestern University, 1973.

[19] P. Marcotte , A. Mercier, G. Savard, and V. Verter, "Toll policies for mitigating hazardous materials transport risk," *Transportation Science, INFORMS*, vol.43, pp. 228-243, 2009.

[20] H. Yang, and M.G.H. Bell, "Transportation bilevel programming problems: Recent methodological advances," *Transportation Research*, Part B, vol.35, pp. 1–4, 2001.

[21] M. Gallo, L. D'Acierno, and B. Montella, "A meta-heuristic approach for solving the Urban Network Design Problem," *European Journal of Operational Research*, vol. 201, no.1, pp. 144–157, 2010.

[22] M. Labbé, P. Marcotte, and G. Savard, "A bilevel model of taxation and its applications to optimal highway pricing," *Management Science*, vol.44, pp.1608–1622, 1998.

[23] V. Kalashnikov, F. Camacho, N. Kalashnikova, R. Askin, "Comparison of Algorithms Solving a Bi-Level Toll Setting Problem," *International Journal of Innovative Computing, Information and Control*, vol.6, pp. 3529 – 3549, 2010.

[24] P. Hansen, Y. Kochetov, and N. Mladenovic, "Lower bounds for the uncapacitated facility location problem with user preferences," Tech. Rep. G-2004-24, GERAD-HEC, Montreal, Canada, 2004.

[25] H.A. Eiselt, and G. Laporte, "Sequential location problems," *European Journal of Operational Research* vol. 96, pp. 217-231, 1996.

[26] H.I. Calvete, C. Galé, and M.J. Oliveros, "Bilevel model for production-distribution planning solved by using ant colony optimization," *Computers & Operations Research*, vol.38, pp. 320-327, 2011.

[27] L. Vicente, and H. Calamai, "Bilevel and multilevel programming: A bibliography review," *Journal of Global Optimization*, vol.5, no.3, pp. 291-306, 1994.

[28] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of Operations Research*, vol.153, pp. 235-256, 2007.

[29] R. Mathieu, L. Pittard, and G. Anandalingam, "Genetic algorithm based approach to bilevel linear programming, R.A.I.R.O.," *Recherche Operationelle*, vol.28, pp. 1-21, 1994.

[30] V. Oduguwa, and R. Roy, Bilevel Optimisation Using Genetic Algorithm, in *IEEE International Conference on Artificial Intelligence Systems (ICAIS'02)*, pp.32, 2002.

[31] J. Bhadury, J. Jaramillo, and R. Batta, "On the Use of Genetic Algorithms for Location problems," *Computers & Operations Research,* vol. 29, pp. 761-779, 2002.

[32] J. Yang, M. Zhang, B. He, and C. Yang, "Bi-level programing model and hybrid genetic algorithm for flow interception problem with customer choice," *Computers & Mathematics with Applications*, vol.57, no.11, pp. 1985-1994, 2009.

[33] E. Alekseeva, N. Kochetova, Y. Kochetov, and A. Plyasunov, "A hybrid memetic algorithm for the competitive *p*-median problem," in *Preprints of the 13th IFAC Symposium on Information Control Problems in Manufacturing*, pp. 1516-1520, 2009.

[34] I.L. Vasilyev, and 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, 2009.

[35] H. Kücükaydin, N. Aras, and K. Altinel, "A hybrid Tabu Search Heuristic for a Bilevel Competitive Facility Location Model," *Lectures Notes in Computer Science*, vol. 6373, pp.31-45, 2010.

[36] F. Legillon, and A. Liefooghe, E.G. Talbi, CoBRA: A Coevolutionary Meta-heuristic for Bi-level Optimization, in *IEEE Congress on Evolutionary Computation (CEC 2012),* 2012.

[37] L. Brotcorne, F. Cirinei, P. Marcotte, and G. Savard, "A tabu search algorithm for the network pricing problem," *Computers & Operations Research*, vol. 39, no.11, pp. 2603-2611, 2012.

[38] 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*, vol. 2014, 14 pages, 2014.

[39] J. Nash, "Non-Cooperative Games," *The Annals of Mathematics*, Second Series, vol.54, no.2, pp.286-295, 1951.

[40] Stackelberg, H.v., Marktform und Gleichgewicht, *Springer-Verlag, Berlin*, 1934. engl. transl: "The Theory of the Market Economy", *Oxford University Press*. 1952.

[41] J.F. Wang, and J. Periaux, "Multi-Point Optimization using Gas and Nash/Stackleberg Games for High Lift Multi-airfoil Design in Aerodynamics," in *IEEE Proceedings of the 2001 Congress on Evolutionary Computation*, vol.1, pp. 552-559, 2001.

[42] C.O. Pieume, L.P. Fotso, and P. Siarry, "Solving bilevel programming problems with multicriterio optimization techniques", *OPSEARCH*, vol. 46, no. 2, pp. 169-183, 2009.

[43] H.I. Calvete, and C. Galé, A Multiobjective Bilevel Program for Production-Distribution Planning in a Supply Chain, In: Ehrgot, M., et. al. (Eds.), Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, *Lecture Notes in Economics and Mathematical Systems*, 634. Springer-Verlag, pp. 155-165, 2010.

[44] W. Candler, "A linear bilevel programming algorithm: A comment". *Computers & Operations Research*, vol.15, pp. 297–298, 1988.

[45] P. Clarke, A. Westerberg, "A note on the optimality conditions for the bilevel programming problem," *Naval Research Logistics Quaterly*, vol.35, pp. 413–418, 1988.

[46] A. Haurie, G. Savard, and D. White, "A note on: an efficient point algorithm for a linear two-stage optimization problem," *Operations Research*, vol.38, pp.553-555, 1990.

[47] P. Marcotte, and G. Savard, "A note on the Pareto optimality of solutions to the linear bilevel programming problem," *Computers & Operations Research*, vol.18, pp. 355-359, 1991.

[48] J.R. Kim, J.B. Jo, and H. Yang, A Solution for bi-level network design problem through Nash genetic algorithm, in: Szczuka, M., Howard, D., Slezak, D., Kim, H., Kim, T., Ko, I., Lee, G., Sloot, P. (Eds.), Advances in Hybrid Information Technology, *Lecture Notes in Computer Science*, vol. 4413. Springer, Berlin, pp. 269–280, 2007.

[49] C.S. Fisk, "Game Theory and Transportation System Modelling", *Transportation Research Part B*, vol. 18, pp. 301-313, 1984.

[50] T.L. Friesz and P.T. Harker, "Freight network equilibrium: a review of the state of the art", in: Daughety, AF. (Ed.), *Analytical studies in transportation economics*, Cambridge University Press, New York.

[51] J. Moreno, "Metaheurísticas: Concepto y propiedades", Departamento de Estadística, I.O. y Computación, Universidad de La Laguna, 2004.