UNIVERSIDAD MAYOR DE SAN ANDRÉS
Facultad de Ciencias Puras y Naturales
Carrera de Informática

TESIS DE GRADO

Tecnología Evolutiva Aplicada al Diseño de Redes de Comunicación Confiables

Autor: Iván Rodrigo Rivera Tola
Tutor: Lic. Mario Loayza Molina
Revisor: Lic. Rubén Alcón López

La Paz – Bolivia
2009

Para:

Quienes son sin duda la parte más importante de mi vida, a mis padres y hermanos, quienes estuvieron a mi lado en este recorrido…

AGRADECIMIENTOS

Ante todo a mis padres pues sin su apoyo nada de esto hubiera sido posible, gracias por su apoyo y su paciencia, gracias por su ayuda y enseñanzas.

Mis agradecimientos al Lic. Rubén Alcón, por su tiempo y paciencia en la revisión de este trabajo, gracias por su apoyo, sus consejos y orientación durante todo el desarrollo del mismo.

Agradezco también al Lic. Mario Loayza, por su guía y colaboración en el seguimiento de este trabajo.

A mis amigos, que me ofrecieron no solo su amistad también su apoyo y ayuda.

Y finalmente a la Universidad Mayor de San Andrés, la Carrera de informática y todo el personal docente que contribuyeron a mi formación profesional.

Muchas gracias a todos!!!

Iván Rodrigo Rivera Tola

RESUMEN

En los últimos años las redes de comunicación han adquirido una enorme importancia en una gran variedad de aplicaciones. Esto ha producido un rápido desarrollo de la infraestructura de redes, del software y de los servicios, lo que ha renovado el interés por los problemas de diseño de redes, generando un enorme desafío, pues son necesarias redes cada vez más complejas, más confiables, más eficientes y más económicas.

El continuo crecimiento en el tamaño de los problemas ha llevado a los investigadores a proponer alternativas a los enfoques exactos tradicionales para la solución de problemas complejos. En este contexto varias técnicas y heurísticas se han aplicado al problema de diseño de redes. De estas técnicas los algoritmos genéticos AG han experimentado un notable adelanto como alternativa a las técnicas clásicas, los AG son muy atractivos como herramientas para tareas de optimización, se cree que los AG pueden encontrar soluciones mucho mejores en tiempos más cortos y dado que trabajan al nivel de la codificación, es difícil de que obtengan resultados engañosos aún cuando la función pueda ser muy difícil para los esquemas tradicionales.

De lo señalado más arriba y considerando una red de comunicaciones, el problema de Steiner generalizado GSP, se refiere al diseño de una red de mínimo costo que verifique ciertos requerimientos prefijados de conexión. Con respecto a otros problemas de diseño, el GSP incorpora requisitos adicionales de conectividad para garantizar la alta confiabilidad, haciendo al funcionamiento de la red más tolerante y resistente a fallas de sus componentes.

El presente trabajo, propone estudiar y resolver el problema de diseño óptimo de redes, tomando en cuenta para esta tarea, distintos criterios de optimización: diseño de redes de mínimo costo y diseño de redes de máxima confiabilidad; esto a través de la resolución del Problema de Steiner mediante tecnologías evolutivas. No obstante, la metodología propuesta permite abordar distintos problemas de diseño con otros criterios de optimización, por ejemplo la confiabilidad con restricciones económicas, puede ser resuelta con las herramientas presentadas en este trabajo mediante la adaptación de la función objetivo.

ÍNDICE GENERAL

ÍNDICE ESPECÍFICO

CAPITULO 1

INTRODUCCIÓN

1. INTRODUCCIÓN

RESUMEN

En este capítulo se describe y plantea el problema de la optimización topológica, se muestra las dificultades que conlleva su resolución y se propone la forma de resolverlo mediante el aprovechamiento de tecnologías evolutivas; se establecen los objetivos, se plantea la hipótesis y se especifica la justificación de la investigación; además se definen los alcances y limitaciones del objeto de estudio y finalmente la metodología para la demostración de la hipótesis.

En los últimos años las redes de comunicaciones han adquirido una enorme importancia en una gran variedad de aplicaciones. Esto ha producido un rápido desarrollo de la infraestructura de redes, del software y de los servicios de Internet, lo que ha renovado el interés por los problemas de diseño de redes de comunicaciones, generando asimismo un enorme desafío para el diseño, pues son necesarias redes cada vez más complejas, más confiables, más eficientes y más económicas.

Uno de los objetivos más importantes de la planificación de sistemas de telecomunicaciones, es el diseño de la configuración necesaria para prestar un servicio de manera óptima respecto de algún criterio de desempeño. Por ejemplo, si el criterio de desempeño es el costo, un problema a resolver es encontrar una topología de red que interconecte sus nodos al menor costo y que tenga la propiedad de asegurar la comunicación confiable de datos.

La optimización topológica en el diseño de redes es un problema clásico de investigación de operaciones y optimización combinatoria, con aplicabilidad práctica en diversos campos como las telecomunicaciones, planificación de redes de computadoras, redes ferroviarias, desagües, gasoductos y oleoductos entre otros (Benjamín Barán, 2000).

El problema de diseño, se plantea como la interconexión de un conjunto de nodos emisores/receptores, geográficamente distribuidos mediante una topología de red de bajo

costo y con capacidad de proveer confiabilidad[1] superior o igual a la mínima especificada en los parámetros de diseño.

Como medida de confiabilidad, se considera una métrica de conectividad conocida como confiabilidad uniforme y definida como la probabilidad de que exista comunicación entre cada par de nodos durante un periodo determinado de tiempo, esto es; que la topología sea al menos un spanningtree [2] o árbol de mínima expansión MST.

El MST tiene una historia respetable en la optimización combinatoria, fue propuesto en primera ocasión por Boruvka en 1926, quien según se dice tuvo que aprender de éste durante su trabajo en la electrificación del sur de Moraria, donde él proporcionó una solución para hallar la distribución más económica a través de una red de una línea de energía (Campos Y., 2002). Desde entonces la formulación del MST ha sido aplicada en numerosos problemas combinatorios tales como: problemas de transporte, diseño de redes de telecomunicaciones, sistemas distribuidos y otros.

Este problema es importante en dos aspectos; Por el lado teórico, pertenece al conjunto de problemas definidos como NP-Hard[3], por lo que dada su complejidad es importante encontrar técnicas estocásticas que encuentren soluciones aceptables en tiempos adecuados, ya que las técnicas determinísticas tienen un comportamiento exponencial y en las técnicas heurísticas conocidas se degrada la calidad de la solución conforme crece el tamaño del problema.

[1] La confiabilidad en una red, es una medida que evalúa la probabilidad de éxito en la comunicación entre pares de nodos

[2] Spanningtree. definido como la probabilidad de que exista comunicación entre cada par de nodos terminales de una red durante un periodo determinado de tiempo

[3] En teoría de la complejidad computacional, la clase de complejidad NP-hard puede ser descrita como la que contiene los problemas de decisión que son al menos tan difíciles como un problema NP.

Por el lado práctico, problemas como encontrar la configuración óptima de redes de cómputo, redes de carreteras, o en general problemas de flujo son equivalentes al problema de MST.

En el diseño topológico de redes, donde se tiene un conjunto de terminales y otras fuentes de datos diseminadas sobre un área geográfica extensa y con alguna medida de tráfico esperado entre varias fuentes, surgen preguntas como: ¿Cuántos concentradores se necesitan? ¿Cómo deben ser conectados? ¿Dónde deben ser colocados?, etc. y a partir de esto se busca determinar que terminales deben ser conectadas a un concentrador en particular.

La gran demanda de comunicaciones de datos en los últimos años ha propulsado el diseño y desarrollo de la infraestructura de redes. El continuo crecimiento en el tamaño de los problemas ha llevado a los investigadores a proponer alternativas a los enfoques exactos tradicionales para la solución de problemas complejos. En este contexto varias heurísticas se han aplicado al problema de diseño de redes de comunicación confiables.

De estas técnicas, los Algoritmos Genéticos **AG** cuentan con un muy buen balance entre la exploración y explotación, ya que aunque el AG esté cercano al punto de convergencia, el operador de cruza nos permite explorar individuos con diferentes características. Por otra parte, el operador de mutación permite al AG una exploración sobre un área determinada del espacio de soluciones.

La capacidad de encontrar soluciones de calidad, eficientes y robustas de los AG es una idea que llama la atención de muchos investigadores para trabajar con ellos, sobre todo porque los métodos tradicionales son buenos para ciertas clases de problemas específicos, pero cuando un problema no se adapta a tales métodos, encontrar la solución al problema puede ser una tarea muy ardua.

Por varias razones, los AG son muy atractivos como herramientas para varias tareas de optimización. Primero, presentan una abstracción de lo material y los operadores genéticos tal y como se presenta en la naturaleza. Segundo, se cree que los AG, como procedimientos de búsqueda multi-punto, pueden encontrar soluciones mucho mejores en tiempos más cortos que los procedimientos clásicos de búsqueda de un punto. Tercero, dado que los AG trabajan al nivel de la codificación, es difícil de que obtengan resultados engañosos aún cuando la función puede ser muy difícil para los esquemas tradicionales. Y cuarto, se piensa que los AG entregan un alto rendimiento, dado que muestrean los esquemas en los cromosomas de una manera inherentemente paralela.

1.1. ANTECEDENTES

Problemas similares al descrito, han sido largamente estudiados utilizando tanto métodos enumerativos, como heurísticos, entre estos se pueden citar: "menor-costo para el diseño de una topología de red[4], aplicación de búsqueda del Tabú"; "Diseño topológico de redes de comunicación de computadoras que usan el diseño simulado" pero también Algoritmos genéticos como en el trabajo "Búsqueda local, Algoritmo Genético para el diseño Óptimo de Redes Fiables" siendo estos últimos los que han demostrado los mejores desempeños (Benjamín Barán, 2000).

En el trabajo realizado por Dengiz, F. Altiparmak y A. Simth, "Efficient optimization of all-terminall reliable networks using an evolutionary approach", IEEE Transaction on Reliability, Vol. 46, 1997. Se presenta un algoritmo genético cuya función objetivo se basa en una minimización de costo, bajo la restricción de confiabilidad uniforme mínima. Dicho trabajo no considera redundancia y se asume que todos los enlaces son de igual confiabilidad (Benjamín Barán, 2000). Más tarde, los mismos autores presentan una versión mejorada, introduciendo operadores genéticos especializados con búsqueda local y reparación.

[4] La topología de red define la estructura de una red; una parte de la definición topológica es la topología física, que es la disposición real de los cables o medios, la otra parte es la topología lógica, que define la forma en que los hosts acceden a los medios para enviar datos.

Posteriormente, D. Deeter y A. Smith, presentan el trabajo "Economic Design of Reliable Networks", IIE Transactions, Special Issue on Economics of Reliability Engineering, 1998. Se extiende el problema paradigma, permitiendo el uso de distintos tipos de enlaces, además de demostrar la versatilidad de los AG para tratar problemas relacionados, como la maximización de confiabilidad bajo restricción de coste máximo, así como el uso de distintas métricas de confiabilidad.

En base a estos estudios y considerando una red de comunicaciones con nodos distinguidos y denominados terminales, el problema de Steiner generalizado, **GSP** por su nombre en inglés "Generalized Steiner Problem", se refiere al diseño de una red de mínimo costo que verifique ciertos requerimientos prefijados de conexión entre pares de nodos terminales.

Con respecto a otros problemas de diseño de redes de comunicaciones, el GSP incorpora requisitos adicionales de conectividad para garantizar la alta confiabilidad, haciendo al funcionamiento de la red más tolerante y resistente a fallas de sus componentes.

Desde el punto de vista algorítmico los Algoritmos Genéticos además, pueden explotar el paralelismo intrínseco del mecanismo evolutivo, trabajando simultáneamente sobre varias poblaciones semi-independientes para resolver el problema, eventuales intercambios de soluciones, introducen la diversidad para evitar problemas de convergencia en óptimos locales poblacionales.

En nuestro medio muy pocos han sido los trabajos que han intentado realizar una investigación y profundización respecto a este problema, en una búsqueda a nivel local, podemos encontrar algunos trabajos realizados en nuestra universidad "UMSA"; que utilizan algoritmos evolutivos en la búsqueda de valores óptimos, pocos han tocado la optimización multiobjetico y ninguno aplicado a la optimización de diseño topológico.

En 2001 Vania Vargas de la Universidad Católica San Pablo escribe el trabajo "Solución de problemas no lineales mediante algoritmos genéticos" que muestra el estudio e implementación de los AG, para la resolución del problema de transporte con carga fija. En su trabajo optimiza rutas, sin embargo no analiza la topología ni la confiabilidad.

En general queda mucho camino más por recorrer en la investigación de los Algoritmos Genéticos y su natural adaptación al problemas no convencionales y mucho más todavía en su aplicación.

1.2. DESCRIPCIÓN DEL PROBLEMA

Las redes de comunicaciones han experimentado un enorme crecimiento en la última década debido, entre otros factores, al uso progresivo de la red de redes o Internet. Los requerimientos de calidad de servicio y confiabilidad de las redes modernas, acompañadas por grandes inversiones en ellas, han tornado críticos los problemas de diseño de las mismas siendo necesario el diseño óptimo de redes que reúnan características determinadas.

Esto conlleva la resolución de problemas cada vez más complejos y más grandes. Complejos desde un punto de vista de complejidad computacional y grande respecto del tamaño de las entradas de instancias de dichos problemas. Desde el conocido problema del Minimum Spanning Tree o MST resuelto a través de un algoritmo determinístico, varios tipos de problemas confluyen cuando se trata de optimizar una red de comunicaciones.

Además, son diferentes los criterios de optimización que pueden ser considerados: el costo de instalación, costo de operación, confiabilidad de la red, o incluso más de un objetivo a optimizar simultáneamente.

La mayoría de los problemas involucrados en diseño de redes están catalogados como problemas de tipo NP-Hard o NP-Completo, es decir que no existen a la fecha algoritmos que puedan encontrar solución en un tiempo acotado polinomialmente respecto del tamaño de la entrada.

No obstante, la necesidad de encontrar soluciones basada en las cuestiones mencionadas anteriormente, han conducido las investigaciones a la utilización de heurísticas y meta heurísticas que permitan, si bien no el óptimo global, encontrar buenas soluciones. Así, algoritmos como simulated annealing, tabú search, redes neuronales y algoritmos evolutivos entre otros han experimentado un notable adelanto e interés por parte de los investigadores como técnicas alternativas a las técnicas clásicas que permiten la obtención de buenas soluciones a los problemas planteados.

El presente trabajo pretende estudiar y resolver uno de estos problemas de diseño de redes utilizando algoritmos evolutivos y tomando en cuenta para esta tarea, distintos criterios de optimización: diseño de redes de mínimo costo y diseño de redes de máxima confiabilidad. En concreto, esta tesis abordará el diseño óptimo de redes de alta confiabilidad a través de la resolución del Problema de Steiner mediante tecnologías evolutivas.

No obstante, la metodología propuesta permite abordar otros problemas de diseño con otros criterios de optimización, por ejemplo la confiabilidad con restricciones económicas, puede ser resuelta con las herramientas presentadas en este trabajo mediante la adaptación de la función objetivo.

Con algún esfuerzo adicional, pueden atacarse problemas con criterios de optimización multi-objetivo; por ejemplo, optimizar simultáneamente costo, instalación y confiabilidad.

1.3. PLANTEAMIENTO DEL PROBLEMA

En el diseño de redes de comunicaciones, una etapa importante consiste en encontrar la mejor forma de interconectar los componentes de una red, optimizando alguna variable y respetando ciertas restricciones impuestas. La variable a optimizar suele ser el costo mientras que las restricciones más comunes están aplicadas sobre flujos de datos, retardos de transmisión y confiabilidad, entre otras.

El problema del diseño de redes óptimas puede representarse a través de un modelo matemático o conjunto de ecuaciones que relaciona las variables involucradas y consiste en determinar una topología definida por la cantidad y posición de los enlaces que transportarán los datos a través de la red y sus características.

A fin de obtener una distribución óptima de enlaces, con respecto a algún criterio específico, se define el Diseño Optimo de la Red de Datos como un problema de optimización, cuya formulación general es la siguiente (Magnago, 2006):

$$Min / Max\ f(\boldsymbol{q})$$
$$y \quad g(\boldsymbol{q}) \leq g^*$$

Fórmula (1.1) (Magnago, 2006)

Siendo g* el valor máximo admitido y q es un vector de variables binarias tal que:

$$\begin{cases} q_i = 1, & si\ el\ enlace\ i\ existe \\ q_i = 0, & en\ caso\ contrario \end{cases}$$

Los criterios de desempeño $f(\boldsymbol{q})$, pueden ser diversos, por ejemplo: de tipo económico, relacionados con la capacidad del enlace, los retardos, la confiabilidad del sistema, etc. En cuanto al conjunto de las restricciones, el diseño debe garantizar también la satisfacción de las condiciones impuestas, en este caso la confiabilidad o disponibilidad.

Cualquiera sea la formulación dada, el problema resultante es un problema de optimización combinatoria. La resolución o forma de ataque de esta clase de problemas, depende fuertemente del caso particular que se trate. Una clasificación de los métodos de resolución propuestos puede ser la siguiente:

i) Diseños basados en la utilización un algoritmo determinístico

Para algunos problemas de diseño, existen algoritmos determinísticos bien probados y bien conocidos que arrojan resultados satisfactorios. El algoritmo de Dijkstra, permite encontrar el camino de mínimo costo entre dos nodos de una red, para construir las tablas de ruteo en cada nodo, además, garantiza la obtención del óptimo global en tiempo polinomial.

ii) Diseños óptimos utilizando algoritmos estocásticos

Desde el punto de vista de la optimización son métodos probabilísticos de orden cero, es decir que sólo requieren los valores de la función a ser optimizada. La búsqueda del óptimo está guiada por conceptos probabilísticos y pueden resolver cualquier tipo de problemas de optimización. Las desventajas residen en que, dada su naturaleza estocástica, no ofrecen ninguna garantía de convergencia para una corrida dada, y además el costo computacional puede ser elevado; aunque esta última característica puede ser mejorada sustancialmente, incorporando conocimiento específico del problema a resolver

El problema que se plantea en esta tesis es uno de un gran grupo de problemas para los cuales no han podido desarrollarse algoritmos que sean lo suficientemente rápidos para encontrar el óptimo global. Sin embargo es posible diseñar algoritmos eficientes que arrojen una solución cuasi-óptima mediante AG. Estos algoritmos no garantizan la obtención del óptimo global, pero pueden encontrar buenas soluciones, teniendo en cuenta los tiempos de cálculo.

1.3.1. Problema de diseño de redes

Una red puede ser modelada a través de un grafo simple, sin lazos ni redundancias $G = (V, E)$ *donde* $V = \{v_1, v_2, \ldots, v_n\}$ es el conjunto de n nodos o vértices que representan los centros de comunicaciones de la red y $E = \{e_1, e_2, \ldots, e_m\}$ es un conjunto de m vínculos o enlaces que permiten la conexión entre los nodos.

Cada enlace se identifica con un par no ordenado. Los vínculos pueden ser orientados o no-orientados y que representan enlaces de comunicaciones que son unidireccionales o bidireccionales respectivamente, pero no se permiten lazos cerrados ni enlaces redundantes, un par de nodos cualesquiera están directamente vinculados, como máximo, por un único enlace. Un enlace es incidente sobre dos nodos, si los dos nodos están unidos por el enlace.

En la siguiente figura se muestra un ejemplo de un grafo no orientado que representa una red como la descrita:

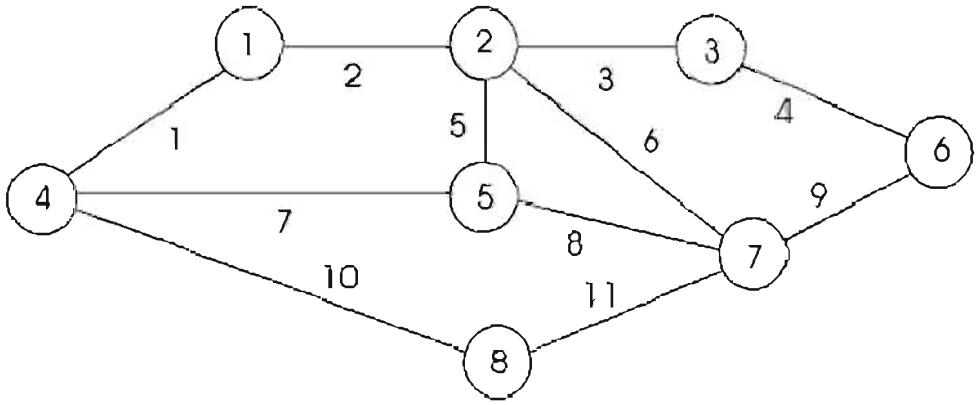


Figura 1.1: Grafo simple en representación de una red (Magnago, 2006)

Donde:

$G = (V, E)$

$V = \{1,2,3,4,5,6,7,8\}$

$E = \{(1,2),(1,4),(2,3),(2,5),(2,7),(3,6),(4,5),(4,8),(5,7),(6,7),(7,8)\}$

La cardinalidad de un conjunto *H*, denotada como |H| es la cantidad de elementos que dicho conjunto posee. Así el número de nodos en un grafo G se representa como $n = |V|$ mientras que el número de enlaces es $e = |E|$. Para el grafo de la figura 1.1 $n = |V| = 8$ y $e = |E| = 11$.

Un problema de diseño de redes consiste en la elección de enlaces de comunicaciones de diferentes características o tecnologías entre un conjunto de nodos dado, de forma tal que la red resultante alcance un conjunto de valores, tales como el costo o la confiabilidad y que cumpla con otro conjunto de requerimientos que operan como restricciones al problema.

Existen dos aspectos a tener en cuenta en el diseño óptimo de redes confiables: la obtención de la topología de mínimo costo y el cálculo de la confiabilidad. El segundo aspecto limita el espacio de búsqueda factible entre el conjunto de todas las topologías posibles. Además la confiabilidad depende de la topología de la red.

La minimización del costo total de una red de comunicaciones y la maximización de su confiabilidad son objetivos contrapuestos pero necesarios e ineludibles. Por ello, un modelo que minimice el costo total de la red, satisfaciendo los requisitos de conexión entre todo par de nodos terminales sin agregar redundancia de caminos, constituye una solución muy sensible y propensa a fallas de conexión en los nodos o en los enlaces que afectarían a la operativa de la red.

El Problema Generalizado de Steiner GSP (Nesmachnow, 2002), (Magnago, 2006), (Benjamín Barán, 2000), (Campos Y., 2002), incorpora requisitos adicionales de conectividad sobre cualquier par de nodos terminales, aplicándose al diseño de redes de comunicaciones en las cuales la confiabilidad queda garantizada por la existencia de diferentes caminos alternativos entre estos pares de nodos terminales, haciendo el funcionamiento de la red más resistente a fallas de sus componentes y por ende más segura y aumentando su nivel de disponibilidad.

En el diseño de la red se considera cualquier tipo de enlace independientemente de su tecnología, así como también redundancia.

El planteamiento se denota de la siguiente manera:

Una red de comunicaciones puede ser modelada mediante un grafo probabilístico G={V,E}, donde V representa un conjunto de nodos y E un conjunto de enlaces. Entre cada par de nodos i,j que pertenecen a V, pueden existir distintos tipos de conexiones l_{ijk} que pertenecen a E, donde k que pertenece al conjunto K, representa la tecnología utilizada. Cada enlace l_{ijk} tiene un costo $C(l_{ijk})$ y una confiabilidad $p(l_{ijk})$ que pueden diferir de un enlace a otro (Benjamín Barán, 2000).

El problema de optimización consiste básicamente en la minimización de la expresión:

$$\sum_{i=1}^{|V|-\ |V|} \sum_{j=i} \sum_{k=i}^{K} c(l_{ijk}) x_{ijk}$$

Sujeto a R(X) ≥ Ro

Fórmula 1.2 (Benjamín Barán, 2000)

Donde x_{ijk} es una variable de decisión que pertenece al intervalo {0,1} que indica la existencia o no de un enlace entre un par de nodos de la red.

R(X) es la confiabilidad uniforme, definida por el número de enlaces redundantes entre cada par de nodos terminales de la topología candidata: $\mathbf{X} = \{x_{ijk}\}$.

Y, **Ro** es la confiabilidad mínima requerida para el diseño de la red

1.4. OBJETIVOS

En el presente trabajo se plantean los siguientes Objetivos:

1.4.1. General

Resolver el problema generalizado de Steiner para el diseño topológico de redes de comunicación de computadoras de tal manera que este diseño sea confiable y de bajo costo, en base a la aplicación de una estrategia evolutiva.

1.4.2. Específicos

Se detallan a continuación los objetivos específicos que se pretenden lograr con el trabajo de investigación:

- Estudiar las diferentes alternativas para solucionar problemas de tipo combinatorio relacionados con el objetivo principal planteado, haciendo énfasis en las técnicas que aportan las tecnologías evolutivas.
- Aplicar una técnica evolutiva que logre resolver el problema de Steiner en un grafo no orientado que representa geográficamente la topología de una red.
- Interconectar un conjunto de nodos, geográficamente distribuidos, mediante una topología de red de bajo costo con capacidad de proveer una confiabilidad superior a la mínima especificada en los parámetros de diseño.
- Analizar los resultados del AG en comparación con los otros métodos

1.5. HIPÓTESIS

Hi: *"Es posible resolver el problema generalizado de Steiner para el diseño de una red de comunicación confiable y de bajo costo representada por un grafo probabilístico, mediante una técnica evolutiva como método eficiente de diseño"*

1.5.1. Descripción Informal

En el modelo experimental que se pretende desarrollar, se establece al AG como un elemento que servirá para la búsqueda de una solución optima en el diseño topológico de una red, es decir la solución del problema de Steiner de k conexión.

1.5.2. Descripción de Variables

La identificación de variables participantes es como sigue: sea VI(x) la variable independiente, VD(y) la variable dependiente, VIV(z) la variable interviniente y VM(w) la variable moderante, entonces decimos que:

$VI_1(x)$ = Maximización de la confiablidad
$VI_2(x)$ = Minimización de costo
VD(y) = Sistema para el diseño de red
VIV(z) = Algoritmos genéticos
VM(w) = Topología de red

1.6. JUSTIFICACIÓN

En base a la pregunta de: ¿Qué aportes ofrecerá y para que aplicaciones sería necesario estudiar y establecer un método para el diseño de una red confiable que logre satisfacer un problema de k conexión?, plantea las justificaciones siguientes.

1.6.1. Científica

Tanto la resolución del problema de Steiner como la Optimización Topológica de redes es un problema largamente estudiado por muchos investigadores en el área de la optimización combinatoria; se han propuesto muchas soluciones que si bien resuelven el problema cuando el campo de exploración crece ya no son óptimos, frente a esto los algoritmos genéticos y las técnicas evolutivas ofrecen una nueva alternativa de solución.

Los Algoritmos Genéticos abren las puertas para resolver problemas donde los métodos tradicionales ya no son efectivos y donde no hay una línea definida o procedimientos exclusivos para la resolución de los mismos, amplían el campo de la informática apoyándose en la Biología y Ecología, impulsan además el desarrollo de nuevas herramientas para la ciencia de la computación.

Los AG son una nueva forma de encarar los problemas y su desarrollo va en aumento. En este sentido su aporte a otras áreas científicas también va creciendo y por su esencia natural van evolucionando.

1.6.2. Social

A nivel internacional se han realizado varias investigaciones dentro de esta línea, sin embargo en el área local ninguna o muy pocas. Nuestro medio, en la actualidad exige que podamos presentar soluciones rápidas de bajo costo y precisas, por tanto es de interés ofrecer alternativas y herramientas para la toma de decisiones cuando estamos construyendo una red que exija los niveles de confiabilidad mínimos y que reduzca los costos que su implementación requiera.

Dotar de un medio rápido que optimice los recursos computacionales y además se oriente al diseño de manera confiable, es útil y brinda muchos beneficios en el emprendimiento de una nueva red, sobre todo si se necesita altos niveles de disponibilidad.

Aplicaciones como la telefonía, redes de datos, redes eléctricas, redes ferroviarias, redes de acueductos o lo que sea que refiera una red que deba satisfacer ciertos niveles de confiabilidad y que no ponga en riesgo la actividad por la que fue desarrollada la red se beneficiarán de esta investigación.

1.6.3. Económica

El aporte económico es desde todo punto de vista considerable, reduce los tiempos de cómputo que se traducen en un factor decisivo a la hora de la toma de decisiones, pero sobre todo reduce los costos de diseño final en la topología de la red resultante, minimizando el número de en laces necesarios sin descuidar el tema de la confiabilidad o disponibilidad.

Básicamente la investigación y en específico el problema que se está estudiando por ser en su naturaleza referido a la optimización de rutas y minimización de costos tiene un impacto directo en lo económico.

1.7. APORTES

Este trabajo en base a los lineamientos descritos más arriba, pretende aportar en la investigación de operaciones, la optimización combinatoria, la ampliación del conocimiento de los Algoritmos Genéticos, la aplicación de los mismos y la exploración de métodos de procesamiento.

Además aportar con una técnica evolutiva para el diseño de redes de comunicación que resolverá el problema de Steiner mediante un algoritmo genético y así dotar de una herramienta que ayude a los administradores a mejorar la calidad de sus redes, minimice los costos de su implementación y aumente la confiabilidad y disponibilidad de la red resultante.

Al hablar de confiabilidad no hay forma de eludir el tema de la seguridad, este trabajo no solo ayudará en el diseño de la topología de la red sino que ayudara a mejorar la seguridad de la misma ya que apoya directamente a la disponibilidad de la red, base fundamental de la seguridad.

Desde luego es un aporte para todo tipo de optimización de recursos, desde los computacionales, económicos hasta los de análisis.

En un futuro esta investigación anhela ser un modelo de referencia para la aplicación de AG's en el diseño de redes optimas de todo ámbito desde las telecomunicaciones y el diseño de circuitos electrónicos hasta las carreteras y líneas ferroviarias.

1.8. ALCANCES Y LIMITACIONES

La formulación del Problema de Steiner Generalizado, se basa en la definición de Kraup (Sergio Nesmachnow M. P., 2003). Con las siguientes consideraciones:

- Un grafo no dirigido $G = (V, E)$, con una matriz de costos C asociada a sus aristas.
- Un conjunto distinguido de Nodos terminales T, de cardinalidad $n_t = |T|$ tal que $2 \leq n_t \leq n$, siendo $n = |V|$, la cardinalidad del conjunto de vértices del grafo G.
- Una matriz $R = \{r_{ij}\}$ con i, j $\in$ T, de dimensión $n_T \times n_T$, cuyos elementos son enteros positivos que indican los requerimientos de conectividad o cantidad de caminos, entre todo par de nodos terminales.

El trabajo propone encontrar un sub-grafo G_T de G de costo mínimo tal que para todo par de nodos i, j $\in$ T, $i \neq j$ estos sean r_{ij} arista conexos. Es decir, que existan r_{ij} caminos disjuntos, que no comparten aristas entre los nodos terminales i, j.

Sobre los nodos no pertenecientes al conjunto de nodos terminales no se plantean requisitos de conectividad. Estos nodos llamados Nodos de Steiner, pueden formar parte o no de la solución optima, de acuerdo a la conveniencia de utilizarlos o no.

Para el diseño de la red de comunicación se asume que (Benjamín Barán, 2000):

- Los nodos son bidireccionales
- Los nodos son perfectamente confiables
- Los enlaces fallan en forma independiente unos de otros
- No se consideran reparaciones; y
- No se considera la replicación de un mismo enlace; esto es, entre cada par de nodos i-j a lo más existe un enlace del tipo k. Se adopta esta estrategia debido a que normalmente, cuando más de un enlace de un mismo tipo une dos puntos, los mismos se extienden juntos compartiendo una misma infraestructura.

  Por ejemplo, el costo de tender una fibra óptica a través de un ducto puede ser de 10 $/m, mientras que pasar dos fibras costaría menos de 14 $/m, ya que las mismas compartirían el costo del ducto. En todo caso, se puede representar a cada número de replicaciones de enlaces como un tipo de enlaces distinto (k=1, una fibra; k=2, cinco fibras; etc.).

1.9. METODOLOGÍA

La metodología que se utilizará para demostrar la hipótesis planteada, se basa en el método científico de Mario Bunge que sigue los siguientes pasos:

i) Planteamiento del Problema

El problema del diseño de redes confiables involucra optimizar la topología y minimizar los costos con un nivel de confiabilidad k. Los métodos tradicionales, las limitaciones tecnológicas de las computadoras y el tamaño del espacio de búsqueda hacen que este sea todavía un problema complejo

ii) Especificación de la Hipótesis

Es posible apoyar a la resolución del problema de Steiner de k conexión para el diseño de una red confiable y de bajo costo representada por un grafo probabilístico, mediante un Algoritmo Genético Paralelo aplicado sobre un clúster de computadoras como método eficiente de diseño

iii) Análisis y verificación mediante prototipo y casos de prueba

Para probar el rendimiento del algoritmo evolutivo que resolverá el problema de Steiner para el diseño de redes confiables; se desarrollara un prototipo sobre una red de computadoras que conformaran un clúster.

iv) Verificación de los Resultados

Se pretende evaluar los resultados obtenidos con el modelo mediante el planteamiento de algunos casos de prueba, versus los resultados obtenidos en base al prototipo construido

v) Conclusiones

Finalmente se mencionará las conclusiones obtenidas, estado de la hipótesis y recomendaciones en base a lo obtenido con este estudio

CAPITULO 2

REDES CONFIABLES Y EL PROBLEMA DE STEINER

2. OPTIMIZACIÓN EN EL DISEÑO DE REDES

RESUMEN

Este capítulo describe el problema de optimización en el diseño de redes, se describen las maneras de estimación de su confiabilidad, así como también los métodos y técnicas que se emplean para su resolución. Por otro lado, se describe y se formula el modelo matemático del problema generalizado de Steiner para la optimización en el diseño de redes confiables.

Los problemas de optimización pueden dividirse en forma natural, en dos categorías: aquellos con variables continuas, y aquellos con variables discretas o problemas combinatorios (Magnago, 2006). En los primeros, la atención se centra en un conjunto de números reales o de funciones reales; en los segundos, la atención está puesta en un objeto de un conjunto finito, tal como un entero, una permutación, una combinación o un grafo.

Estas dos clases de problemas tienen características diferentes, y los métodos de resolución aplicados a ellos resultan muy diferentes.

Los problemas de diseño de redes confiables, pertenecen a la segunda categoría. El problema de diseño de red, estudiado en este trabajo, consiste en encontrar una red de comunicación que minimice el costo total de la misma, bajo ciertas condiciones de confiabilidad dadas, específicamente resolviendo el problema planteado por Steiner.

En este contexto se analiza en profundidad un problema de optimización discreta restringido. Es optimización porque el objetivo es minimizar el costo asociado a una red de telecomunicaciones, es discreta porque las variables sólo toman valores de un conjunto finito de números, lo que permite su simulación exacta; finalmente, es restringido debido a que el objetivo está condicionado a cumplir con un conjunto de exigencias. En este caso la restricción es la confiabilidad

Tanto el problema de diseño de redes de comunicación confiables como el cálculo de la confiabilidad de una red dada, son problemas NP-hard. Esto significa que el esfuerzo computacional crece exponencialmente con el número de nodos y enlaces en la red y dado que, durante el proceso de diseño debe calcularse la confiabilidad de un gran número de topologías candidatas, el tiempo de cálculo resulta alto aún con redes de tamaño relativamente pequeño.

Una instancia de un problema de optimización es un par (F,c), donde F es el conjunto de puntos factibles, y c es la función de costo o función objetivo.

Una aplicación tal que: $c: F \rightarrow R^1$ Fórmula 2.1 (Magnago, 2006)

El problema es encontrar $f \epsilon F$ para el cual:

$c(f) \leq c(y)$, para todo $y \epsilon F$ Fórmula 2.2 (Magnago, 2006)

El punto f es llamado un óptimo global. Un problema de optimización es un conjunto I de instancias asociadas al mismo.

Informalmente, en una instancia se tienen los "datos de entrada" y suficiente información como para obtener una solución; un problema de optimización es una colección de instancias, todas generadas en forma similar.

2.1. CONFIABILIDAD EN REDES

Debido a la inherente dificultad de calcular la confiabilidad en una red independientemente del tiempo, se emplea un modelo de probabilidad discreto en el análisis de la misma. Esto es, los componentes de la red pueden estar en uno de dos estados: operativo o en falla y, en forma independiente de los estados de los otros componentes.

Entonces, el análisis de confiabilidad del problema es: dada la probabilidad de que cada componente esté operativo, calcular una medida de la confiabilidad de la red. Esta definición de confiabilidad no incluye consideraciones de reparación; en lugar de eso se dice que: dado un tiempo suficientemente extenso t, se define la confiabilidad como la probabilidad que un componente no falle en el tiempo t.

El punto de arranque es una red *G=(E,V),* donde V es un conjunto de nodos o vértices y E es un conjunto de enlaces no dirigidos o aristas. Cuando se estudia el modelo de conectividad de cada $e \in E$ se define p_e como P (esté operativo), que es la confiabilidad de *e.* Este modelo puede servir también para estudios de flujos de camino más corto, asociando una capacidad *Ce* con cada $e \in E$ y se interpreta p_e como la probabilidad de que *e* esté operativo y tiene una capacidad *Ce.* También definimos 1-*pe* como la probabilidad de que *e* falle o tenga una capacidad cero.

Es conveniente para estudios de confiabilidad de redes definir un contexto para un análisis más general. Un "sistema binario estocástico" representa las fallas aleatorias como una función donde sus componentes T pueden tener dos estados: operativo o en falla. Si S es el conjunto de componentes operativos entonces puede definirse Ψ como sigue:

$$\Psi(S) = \begin{cases} 1 \text{ cuando } S \text{ esta activo y sucede } T - S, \text{ El sistema esta operativo} \\ 0 \text{ cuando } S \text{ esta inactivo y sucede } T - S, \text{ El sistema esta en falla} \end{cases}$$

El modelo computacional se puede definir como:

$$R(S,p) = P\,[\Psi(S) = 1]$$

Fórmula 2.3 (Magnago, 2006)

Dado el modelo de red descrito antes, el problema general del análisis de la confiabilidad de una red se da por la pregunta:

¿Cuál es la probabilidad de que una red esté operativa, dada la probabilidad de que cada uno de los enlaces estén operativos?

2.1.1. Análisis de confiabilidad de una red

Un sistema está formado por componentes los cuales pueden ser susceptibles de falla. La falla de un componente cualquiera se da a un tiempo desconocido pero estimable a través de una variable aleatoria T. La figura 2.1 muestra una curva típica de confiabilidad de un componente en función del tiempo. La forma de la curva tiene relación directa con la distribución de probabilidades de falla del componente.

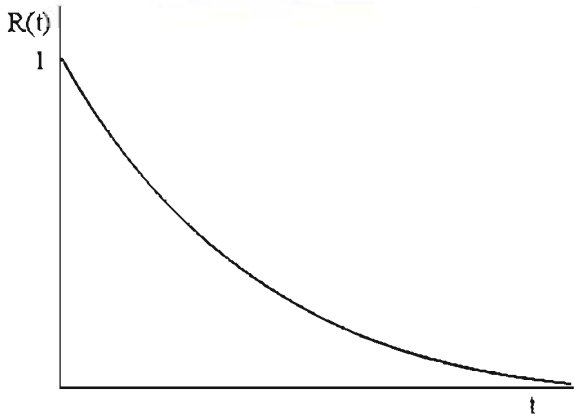


Figura 2.1: Confiabilidad de un componente (Magnago, 2006)

De esta forma la confiabilidad del mismo está dada por la probabilidad que no falle en un tiempo inferior a un valor especificado. Esto es:

$$R = P[\textit{no falla en } T \leq t]$$

Fórmula 2.4 (Magnago, 2006)

Tomando un tiempo t admisible, la confiabilidad del componente será un valor escalar representado por un punto en la curva de la figura 2.1.

Interesa ahora definir una medida o índice de confiabilidad para una red, que esté basada en su topología y en la confiabilidad de sus componentes. La forma de obtener

dichos índices varía dependiendo de los datos con que se cuente. Un dato imprescindible es el referente a la topología de la red, pero según se disponga o no con datos sobre las confiabilidades elementales de los componentes, se pueden adoptar dos enfoques:

- Solidez o resistencia a fallas: Este enfoque se basa en la posibilidad de comparar dos topologías y determinar cuál de ellas es más confiable, para lo cual es necesario considerar algunos parámetros en ellas. Intuitivamente, una red que tiene una topología de árbol "sólo un spanning tree" resulta menos confiable que otra que posee una topología en anillo. Reichelt, propone el conteo de todos los posibles spanning trees en una red para dar un índice de confiabilidad relativa sobre la misma. Otra alternativa es contar la cantidad de puntos de articulación que la red tiene. La topología con menor número de puntos de articulación será más confiable.

- Confiabilidad: Se calcula la probabilidad de funcionamiento de la red sobre la base de las confiabilidades elementales de cada componente, dadas mediante probabilidades de funcionamiento de los mismos.

El enfoque a tomar en este trabajo es el segundo, considerándose un modelo probabilístico basado en un grafo estocástico donde se asignan probabilidades de funcionamiento a nodos y enlaces. Sobre grafos de este tipo, existen básicamente tres medidas de interés:

a) La confiabilidad entre dos nodos, denotada $R_2\,(G)$, es la probabilidad de que para un par específico de nodos s y t, hay al menos un conjunto de enlaces operacionales que conectan los nodos s y t.

b) La confiabilidad de k-nodos, que se denota $R_k\,(G)$, es la probabilidad de que para un conjunto específico K de k nodos, hay un conjunto de enlaces operacionales conectando cada par de nodos en K.

c) La confiabilidad de todos los nodos, R_A *(G)* es la probabilidad que haya un conjunto de enlaces operacionales, o una ruta completa, entre todo par de nodos de la red.

Dado un conjunto de nodos K y un nodo s ∈ K (k = |*K*|) y dada una red G con enlaces de confiabilidad p_e, la confiabilidad de k-nodo está definida por:

$$R(G,s,K,p_e) = P[\text{Existe un camino operativo desde s a cada nodo en K}]$$

De aquí se deducen los dos casos especiales mencionados. Para k = 2, que es el caso de dos terminales, $R_2(G,s,K,p_e)$ y para k=V para todos los nodos, $R_A(G,s,p_e)$. s es el nodo fuente y {K\s} los nodos terminales.

La confiabilidad del sistema dependerá de la confiabilidad de sus componentes y de la interconexión entre ellos. Por lo tanto, la confiabilidad de la red, definida como la probabilidad de que ésta sea operativa, dependerá de las confiabilidades de los enlaces y de la topología de la misma.

En este trabajo se estudia la medida de confiabilidad de todos los nodos modelando a la red mediante un grafo probabilístico, considerando que los enlaces fallan con cierta probabilidad q_e y los nodos son confiables (p_{nodo}=1).

Para un enlace e ∈ E, denotándose con p_e a la probabilidad de que el enlace esté operativo y con $q_e = 1 - p_e$ a la probabilidad que el mismo falle y, si se considera que en cualquier instante sólo un conjunto de enlaces de G estará operacional, un estado de G es un subgrafo (N,E') donde 'E ∈ E es el conjunto de enlaces que está operacional. Si Ω es el conjunto de todos los posibles estados operacionales, la confiabilidad exacta de la red está dada por la siguiente expresión (Dengiz, 1997):

$$R(G) = \sum_{\Omega}[\prod_{e \epsilon E} p_e] * [\prod_{e \epsilon E \setminus E} q_e]$$

Fórmula 2.5 (Magnago, 2006)

El tamaño de Ω crece exponencialmente en función del tamaño de la red haciendo impracticable su cálculo aún tratándose de redes de tamaño moderado.

Se han presentado métodos de cálculo exacto de la confiabilidad basados en enumeración de estados, caminos mínimos "min-path" o conjuntos de corte mínimos "cut-set". No obstante, las mejoras relativas al tiempo computacional no han sido importantes.

Se debe aclarar que el modelo utilizado se adecua a realidades donde la información q debe transmitirse entre dos nodos se rutea de forma dinámica a través de la red. Esto significa que si bien puede existir un camino predeterminado para comunicar dos nodos, se puede elegir un camino alternativo en caso de falla de algún componente intermedio del camino predeterminado; por ejemplo, el ruteo de paquetes entre dos hosts de Internet. No se consideran aquí detalles como el control de congestión y retrasos en la comunicación.

2.1.2. Estimación de la confiabilidad

Debido a los problemas derivados de la complejidad computacional asociada al cálculo exacto de la confiabilidad, se han propuesto métodos que, o bien intentan reducir el espacio explorado o bien, basados en modelos de simulación, permiten estimar un valor para la confiabilidad de una red dada las probabilidades de falla de sus componentes.

Un método ampliamente conocido es el algoritmo de Jan (Jan, 1993), propuesto para encontrar una topología de mínimo costo con una confiabilidad no inferior a un valor dado R0. Jan sugiere calcular un límite inferior de la cantidad de enlaces e*, que

combinados de alguna manera permitan que la red posea una confiabilidad mayor o igual a R0. De esta forma evita la inspección de aquellas redes con una cantidad de enlaces tal que se puede asegurar que su confiabilidad será inferior a la requerida. En realidad, esta propuesta tiende a evitar el cálculo de la confiabilidad cuando a priori puede determinarse que su valor será inferior al requerido y por tanto descartable o penalizable en lo referente al problema de diseño de la red. De hecho, una red de n nodos con menos de n-1 enlaces tiene confiabilidad 0.

La red más confiable con n enlaces debe ser un spanning tree, mientras que si se consideran n+1 enlaces la red más confiable tiene una topología de anillo. Finalmente si se cuentan con n+2 nodos, la red más confiable se dispone como muestra la figura 2.2.

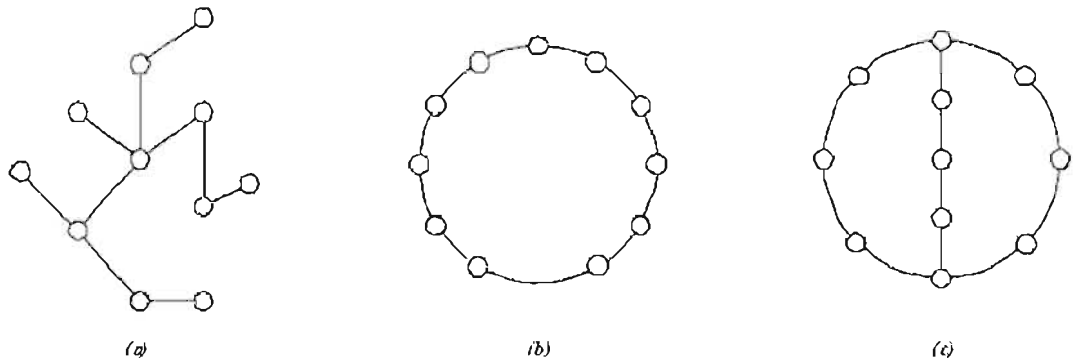


Figura 2.2: Topologías más confiables para una red de 11 nodos (Magnago, 2006)

No obstante, y como se ha mencionado en párrafos anteriores, esta propuesta está orientada a evitar el cálculo cuando a priori puede establecerse que la topología propuesta no alcanza el valor prefijado como admisible. Cuando la cantidad de nodos en la red se incrementa, el tiempo de cómputo nuevamente se hace prohibitivo.

Una alternativa está dada por la aplicación de métodos de simulación para obtener un valor, ahora aproximado, de la confiabilidad.

2.1.3. Evaluación de la confiabilidad por simulación

Los métodos más conocidos de este tipo están basados en la técnica de simulación Monte Carlo, y proporcionan estimaciones con un cierto intervalo de confianza de las medidas de seguridad de funcionamiento. El método Monte Carlo estándar o "Monte Carlo crudo" también sufre de inconvenientes cuando los eventos de interés son raros, pero ocurren con muy baja probabilidad, lo que es común en los sistemas altamente confiables. Es necesario entonces recurrir a otras técnicas, tales como la de la reducción de la varianza, que pueden calcular estimadores más precisos utilizando muestras del mismo tamaño.

El algoritmo Monte Carlo estándar produce resultados aceptables para redes de mediana confiabilidad, pero para redes altamente confiables se deben realizar muchas replicaciones para obtener estados no operativos. Los tiempos de procesamiento crecen linealmente con el número de ensayos, sin mejorar la exactitud en la misma proporción. En pruebas realizadas se observa que con 6000 ensayos se obtienen precisiones del orden de 0,001, por debajo de ese valor la precisión cae notoriamente, por otra parte incrementando el número de ensayo en un orden de magnitud; la varianza mejora muy poco. Estos resultados hacen necesario buscar métodos que equilibren tiempo de procesamiento y precisión.

2.2. DISEÑO OPTIMO DE REDES CONFIABLES

Una vez que hemos decidido la forma de asignar un valor de confiabilidad a una red debemos abordar a continuación el problema de diseño óptimo de redes confiables. Esto quiere decir, determinar para una determinada cantidad y ubicación de nodos a interconectar, una topología que tenga el mínimo costo posible y que a la vez posea una confiabilidad no inferior a un valor prefijado como admisible.

Todo grafo conectado tiene al menos n-1 enlaces activos y $n_{max} = n(n-1)/2$ enlaces posibles. Pueden existir problemas en los cuales no todos los enlaces están disponibles.

Esto reduce la cantidad de enlaces activos posibles a un valor $n^* \leq n_{max}$. Entonces, podrían encontrarse todas las combinaciones posibles de enlaces activos, descomponiendo el problema en $n^*-(n-1)+1 = n^*-n+2$ subproblemas de manera tal que cada uno de ellos involucre una cantidad fija de estos enlaces activos.

La solución al problema principal menos elaborada consistiría en encontrar la mejor solución de cada subproblema para luego optar por la mejor de todas. Sin embargo, este enfoque puede resultar ineficiente desde el punto de vista de procesamiento, porque es necesario resolver todos y cada uno de los subproblemas.

El trabajo de Jan sugiere calcular primeramente un límite inferior de cantidad de enlaces a*, que combinados de alguna manera permitan que la red posea una confiabilidad mayor o igual a la admisible R0. De esta forma se evita inspeccionar aquellos subproblemas cuya cantidad de enlaces se encuentre entre n-1 y a*.

Luego se resuelve el subproblema correspondiente a a*, sp (a*), donde se busca la mejor solución posible con a* enlaces activos y se establece temporalmente la solución encontrada como la mejor solución global.

Seguidamente se procede a evaluar el menor costo esperado para el siguiente nivel (a*+1), mediante la suma de los costos de los a*+1 enlaces más baratos. Si esta combinación es más cara que la ya encontrada en el nivel a*, aquella era la mejor de todas las posibles. En caso contrario, habrá que resolver el subproblema sp(a*+1). Se procede así sucesivamente hasta encontrar la solución del problema principal. De esta manera es posible evitar el registro de cierta cantidad de niveles, dependiendo de las características de cada caso.

Sin embargo existen muchos casos en los cuales el algoritmo presenta problemas que dependen de la rama del árbol que inspecciona. Además, la función que fija el límite superior para la confiabilidad no resulta todo lo eficiente que se quisiera porque en

general ni siquiera es capaz de posicionarse en el subproblema que contiene la solución óptima. Alcanzarlo puede significar la exploración completa de muchos de los niveles anteriores, haciendo que los tiempos se vuelvan nuevamente impracticables en dichos casos. Incluso ya en el subproblema correcto, este límite superior no logra la precisión necesaria como para evitar, en general, el cálculo de la restricción sobre topologías poco confiables.

Si bien estas apreciaciones justifican investigar otras formas de búsqueda, algunas de las estrategias mencionadas hasta aquí son factibles de ser utilizadas dado que es necesario valerse de técnicas que no inspeccionen todo el espacio de búsqueda. Por eso muchas veces se han utilizado heurísticas que permiten una exploración parcial del espacio en la resolución de problemas combinatorios. Entre ellas, los Algoritmos Evolutivos AE, han demostrado ser una interesante alternativa dada su relativamente fácil implementación, su robustez y su posible balance, a través de una correcta determinación de sus parámetros, entre la exploración y explotación del espacio de búsqueda.

Koide (2001) considera una extensión del método de Jan con el objetivo de mejorar los tiempos de cómputo en algunos casos y propone otro método para calcular el límite superior para la confiabilidad. No obstante aunque presenta buenas mejoras respecto del algoritmo de Jan, se reportan resultados para redes de unos pocos nodos.

Dengiz y Co. (1997) propusieron la implementación de un Algoritmo Genético para resolver el problema sobre una buena cantidad de instancias con espacios de búsqueda de hasta $2*10^{90}$. Para reducir el tiempo de cálculo requerido se impuso la restricción adicional de que las redes candidatas debían reunir condiciones de biconectividad.

Reichelt y Co (2003) resolvieron un problema de diseño óptimo de redes confiables utilizando heurísticas de reparación de redes no factibles. La propuesta utiliza una

medida de la confiabilidad entre todos los nodos basada en el conteo de spanning trees que posee la red candidata.

2.3. EL PROBLEMA DE STEINER

Considerando una red de comunicaciones con ciertos nodos distinguidos y denominados terminales, el Problema Generalizado de Steiner GSP, se refiere al diseño de una red de mínimo costo que verifique ciertos requerimientos prefijados de conexión entre todo par de nodos terminales.

En los siguientes puntos se presenta el Problema de Steiner y sus variantes, su modelo matemático y su aplicación al diseño de redes de comunicación confiables.

2.3.1. Formulación del GSP

La siguiente formulación del Problema de Steiner, se basa en la definición original de (Kraup, 1978).
Sean los siguientes elementos:

- Un grafo no dirigido $G = (V, E)$, con una matriz de costos C asociada a sus aristas.
- Un conjunto distinguido de Nodos terminales T, de cardinalidad $n_t = |T|$ tal que $2 \leq n_t \leq n$, siendo $n = |V|$, la cardinalidad del conjunto de vértices.
- Una matriz $R = \{r_{ij}\}$ con i, j ∈ T, de dimensión $n_T \times n_T$, cuyos elementos son enteros positivos que indican los requerimientos de conectividad, o cantidad de caminos entre todo par de nodos terminales.

El problema del GSP, propone encontrar un subgrafo G_T de G, de costo mínimo tal que, para todo par de nodos i, j ∈ T, i ≠ j estos sean localmente r_{ij} arista conexos. Es decir que, existen r_{ij} caminos disjuntos, que no comparten aristas entre los nodos terminales i

y j. El GSP, admite una formulación análoga, que exige la existencia de caminos nodo-conexos "caminos que no comparten nodos". Sobre los nodos no pertenecientes al conjunto de nodos terminales no se plantean requisitos de conectividad. Estos nodos, conocidos como nodos de Steiner, pueden formar parte o no de la solución optima, dependiendo de la conveniencia o no de utilizarlos en la solución.

2.3.2. Modelo matemático del GSP

El modelo del GSP arista conexo, define para cada arista (i, j) ∈ E una variable binaria x_{ij} y una variable real y_{ij}^{kl} que denota la utilidad de la arista (i, j) en la dirección de i hacia j en un camino que une los nodos terminales k y l.

El modelo matemático del GSP como un problema de programación lineal entera es:

$$Min \sum_{(i,j)\in E} c_{ij} \cdot x_{ij} \qquad \text{sujeto a:}$$

$$x_{ij} \geq y_{ij}^{kl} + y_{ji}^{kl} \qquad \forall (i,j) \in E, \forall k,l \in T, k \neq l$$

$$\sum_{(k,j)\in E} y_{kj}^{kl} \geq r_{kl} \qquad \forall k,l \in T, k \neq l$$

$$\sum_{(i,l)\in E} y_{il}^{kl} \geq r_{kl} \qquad \forall k,l \in T, k \neq l$$

$$\sum_{(p,j)\in E} y_{pj}^{kl} - \sum_{(i,p)\in E} y_{ip}^{kl} \geq 0 \quad \forall k,l \in T, \forall p \in V \setminus \{k,l\}$$

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

$$y_{ij}^{kl} \geq 0 \qquad \forall i,j : (i,j) \in E, \forall k,l \in T, k \neq l$$

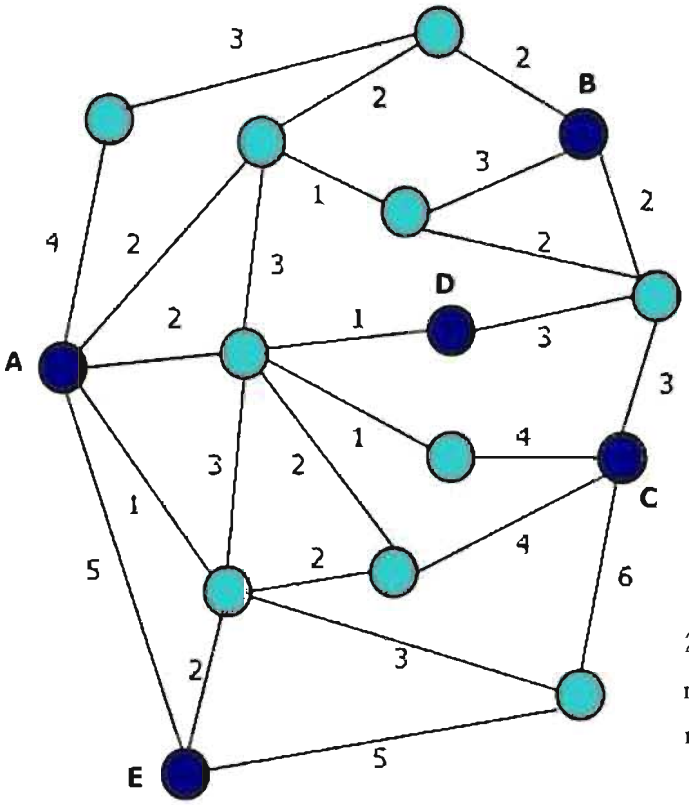
Fórmula 2.6 (Nesmachnow, 2002)

Llamando U={u_{ij}} a la solución óptima del problema, el conjunto de aristas determinado por {(i, j) ∈ E / u_{ij} = 1} define el subgrafo solución G_{SOL}.

2.3.3. Ejemplo de un problema para el GSP

A continuación veamos un ejemplo del GSP sobre un grafo G dado.

Sea el grafo G de la figura 2.3 en cual se ha marcado con un color oscuro los nodos terminales, sobre los cuales se definen los requerimientos de conectividad indicados en la tabla 2.1, mientras que los nodos de Steiner se han marcado con un color claro.



Requerimientos de caminos

Tabla 2.1 requerimiento de caminos sobre los nodos terminales

A	↔	B	3
B	↔	D	2
A	↔	E	1
A	↔	C	3
B	↔	C	3
A	↔	D	2

Figura 2.3 Grafo G para el ejemplo del GSP con requerimiento r_{ij} (Nesmachnow, 2002)

Utilizando la notación del modelo matemático, se tiene que:

$$r_{AB} = 3, \quad r_{AC} = 2, \quad r_{BD} = 1, \quad r_{BC} = 3, \quad r_{AE} = 3, \quad r_{AD} = 2$$

La solución al problema de Steiner planteado sobre el grafo G se muestra en la siguiente figura:

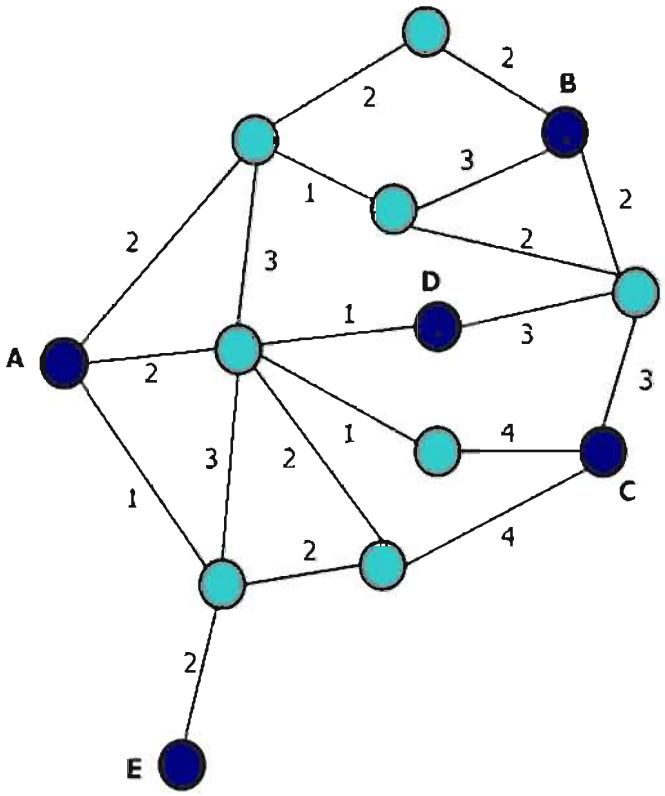


Figura 2.4 Solución al ejemplo del GSP con requerimiento r_{ij} (Nesmachnow, 2002)

2.3.4. Variantes y casos particulares

La complejidad del problema de Steiner obedece a la generalidad de su planteo, al exigir requerimientos variables de conectividad entre pares de nodos terminales que implican la existencia de un número variable de caminos disjuntos en la solución.

Ciertas versiones del problema simplifican su requerimiento, exigiendo una cantidad fija de caminos disjuntos entre pares de nodos terminales.

Esta clase de problemas son conocidos como problemas de *k conexión*, siendo k el número de caminos disjuntos requeridos.

El caso más simple del problema de Steiner es aquel que solo exige un camino entre todo par de nodos (k = 1). La solución de este problema, conocido como problema de *Árbol de Steiner*, corresponde a un árbol de cubrimiento de costo mínimo.

2.3.4.1. Problemas de k conexión

Los problemas de *k* conexión son casos especiales del GSP, en los cuales los requerimientos de cantidad de caminos son constantes para todo par de nodos.

Estos problemas son usuales en aplicaciones donde se desea reducir los costos de enlaces extra entre nodos de la red, manteniendo un numero bajo de caminos disjuntos entre pares de nodos. Como ejemplo, en el diseño de redes de comunicación telefónicas, una solución 2 arista conexa permite introducir el nivel mínimo de redundancia de caminos entre todo par de nodos terminales, limitando la cantidad de enlaces necesarios, los cuales generalmente tienen un costo elevado.

La resolución de un problema de k conexión involucra hallar un subgrafo de cubrimiento k arista conexo de un grafo dado. Por grafo k arista conexo se entiende que no puede convertirse en desconexo removiendo menos de k aristas. Una red de comunicaciones k arista conexa continuara funcionando correctamente, garantizando la comunicación entre todo par de nodos, aun menos k-1 fallos de enlaces.

El problema de k conexión ha sido estudiado bastante, sobre todo el caso k = 2, que propone el diseño de un grafo 2 conexo que cubre un conjunto de nodos terminales con costo mínimo, este además es ampliamente considerado como modelo de redes de comunicaciones y transporte.

El caso genérico del problema de k conexión se define entonces con $k \geq 3$ y es estudiado por varios investigadores.

2.3.4.2. Problema del árbol de Steiner

El problema del árbol de Steiner es un caso particular del problema de k conexión, par el caso k = 1. Ha sido largamente estudiado como un problema clásico de optimización combinatoria.

El requerimiento de conexión simple entre pares de nodos, se soluciona encontrando un árbol de cubrimiento del conjunto distinguido de nodos terminales de costo mínimo.

Existe una diversidad de algoritmos exactos y heurísticas que resuelven el árbol de Steiner, dentro de los algoritmos exactos podemos mencionar los de enumeración de arboles de cubrimiento, los de enumeración de topologías, algoritmos basados en programación dinámica entre otros. Las heurísticas empleadas utilizan técnicas golosas siguiendo caminos de costo mínimo o costo promedio, estrategias de contracción o de cubrimiento.

2.3.5. Complejidad del problema de Steiner

El problema generalizado de Steiner es de tipo NP-Hard tal cual fue demostrado por (Kraup, 1978), tanto para el caso de requerimientos de caminos de aristas disjuntas como para caminos de nodos disjuntos.

El problema de k conexión es también NP-Hard, tal cual se ha probado reduciendo una forma especial del problema a la versión planar del problema en 2 conexiones.

El propio problema del árbol de Steiner, que plantea la restricción menos general en cuanto a caminos, es NP-Hard, de acuerdo a Kraup, inclusive casos más simplificados como las versiones de costos uniformes de las aristas, grafo planar o bipartito en terminales y no terminales, aun son problemas NP-Hard.

Como consecuencia de la complejidad de las versiones del problema de Steiner, la resolución utilizando algoritmos exactos se hace cada vez menos tratable a medida que se aumenta el tamaño del problema. Por este motivo se buscan nuevas alternativas de solución utilizando heurísticas que puedan encontrar estas soluciones en tiempos razonables.

2.3.6. Casos particulares del GSP

En el estudio del problema de Steiner generalizado, ciertos casos particulares son de interés particular, esto en el afán por reducir su complejidad a un orden polinomial. A continuación revisaremos algunos de estos casos:

2.3.6.1. GSP con costos uniformes

Si los costos asociados a las aristas del grafo G son idénticos, el problema GSP se resuelve hallando un subgrafo con mínimo número de aristas que satisfaga los requerimientos r_{ij} de conectividad para los nodos terminales.

(Robledo, 2000) Presenta un algoritmo de orden polinomial para reducir el problema de costos idénticos. Esta solución permite la existencia de aristas paralelas entre nodos. El caso en que no se permiten aristas paralelas en la construcción de la solución ha sido atacado solo en el problema de k conexión, con requerimientos de conectividad uniformes entre nodos.

2.3.6.2. GSP con costos booleanos

El caso del GSP con matriz de costos booleanos. Es conocido como Augmentation Problem, cuya formulación propone hallar el mínimo número de aristas tal que al agregarlas al grafo se satisface los requerimientos de comunicación dadas por una matriz R

2.3.7. Técnicas evolutivas aplicadas a la resolución del GSP

Según aquello que se ha descrito en párrafos anteriores, se puede decir que existen pocos antecedentes de aplicación de algoritmos genéticos al problema generalizado de Steiner.

Sin embargo existen varias referencias a aplicaciones de los algoritmos genéticos a casos particulares como el árbol de Steiner y sus variantes. Las investigaciones datan desde 1989 hasta la fecha con gran variación en los enfoques, cada uno con distintas restricciones para tratar encontrar la mejor solución en tiempos aceptables.

La aplicabilidad de los algoritmos genéticos sobre el problema de Steiner que muestran estas investigaciones, nos dirige a buscar la mejor solución con ayuda de estas técnicas evolutivas, aquellas que reveremos en el capítulo siguiente.

CAPITULO 3

TECNOLOGÍA EVOLUTIVA

3. TECNOLOGÍA EVOLUTIVA

RESUMEN

Este capítulo describe los orígenes y características de los Algoritmos Genéticos, se detalla el funcionamiento de un algoritmo genético simple, sus funciones y métodos, además de sus operadores y subdivisiones. Por otro lado se muestra sus extensiones y el funcionamiento de los algoritmos genéticos paralelos.

En la década de los 60 el investigador John Holland estudió el diseño de sistemas artificiales basados en los mecanismos de los sistemas naturales, e ideó los algoritmos genéticos que posteriormente fueron desarrollados por su grupo de investigación en la Universidad de Michigan (Holland, 1999).

El Algoritmo Genético de Holland es un método para cambiar de una población de individuos que representan posibles soluciones y que a la vez son representados por "cromosomas[5]" a una nueva población. Para este cometido se aplica una especie de "selección natural", donde el elemento a buscar es el individuo mejor adaptado al medio; y unos operadores que se inspiran en la evolución natural, denominados cruce y mutación.

Cada cromosoma está compuesto de genes "bits", donde cada gen puede tener un determinado alelo, valor que para la representación binaria es 0 o 1. La selección se encarga de escoger los individuos que participaran en la formación de la nueva población. El operador de cruce intercambia las propiedades de los individuos y el de mutación produce cambios aleatorios en la población de tal forma que se amplíe el espacio de búsqueda. Holland también introdujo el operador de inversión que invierte el orden de una parte del cromosoma. Al mismo tiempo que Holland desarrollaba los AG, los mismos conceptos fueron utilizados en Alemania por otros investigadores; Rechemberg y Schwefel para desarrollar las estrategias evolutivas (Goldberg, 1989).

[5]En computación evolutiva, el conjunto de genes que representan mediante un alfabeto dado una posible solución al problema que se está resolviendo

La forma original de las estrategias de evolución utilizaba dos individuos y el operador de mutación como único elemento de búsqueda y cambio de la población. Los dos individuos eran el padre y el descendiente. Posteriormente se realizaron nuevas implementaciones de estrategias evolutivas para poblaciones con múltiples individuos.

Por último, la programación evolutiva o genética representa soluciones a los problemas mediante una población de maquinas de estados finitos o mediante programas. La nueva población se crea mediante la mutación aleatoria de los programas padre.

3.1. ALGORIMOS GENÉTICOS SIMPLES

Los algoritmos genéticos simples están basados en un principio básico de la evolución[6]: los mejores individuos tienen una mayor probabilidad de reproducirse y sobrevivir que otros individuos menos adaptados a su medio ambiente (Holland, 1999) y (Goldberg, 1989). Para implementar este principio, los algoritmos genéticos mantienen una población que va evolucionando a través del tiempo y que al final converge a una única solución.

Los individuos de la población se representan mediante un cromosoma que codifica las variables del problema que se quiere resolver. Normalmente se utiliza un cromosoma simple compuesto por una cadena de bits de longitud fija, pero existen algoritmos genéticos que codifican el problema utilizando varios cromosomas para representar una solución y otros que utilizan cromosomas de longitud variable. Cada individuo tiene asignado un valor de una función de coste, que mide la calidad de la solución y que es la herramienta principal que permite simular el concepto de individuos mejor adaptados.

[6]La teoría del origen de las especies mediante selección natural fue propuesta por Darwin, los AG se basan esta teoría para hacer una evolución análoga de los individuos de una población, que representan a posibles soluciones de un problema

Aquellos individuos cuyo valor de la función de coste sea mejor tendrán más posibilidades de ser seleccionados para construir la siguiente población y por lo tanto, para pasar sus propiedades a los individuos de la siguiente generación los cuales a su vez pueden sufrir mutaciones.

Para implementar un algoritmo genético es necesario definir:

- Una función de coste que evalúe a los individuos "fitness".
- Una codificación que permita representar las soluciones.
- Los operadores de selección, cruce y mutación
- El tamaño de la población.
- Los valores de las probabilidades con la que se aplican cada uno de los operadores.

Para comenzar el algoritmo se genera una población inicial a partir de la que se trabaja; cuando se han obtenido los individuos iniciales, se repite el proceso que se detalla a continuación tantas veces como indique la condición de parada; esta condición puede ser un número máximo de generaciones, la convergencia de la población o algún otro específico para el problema que se está resolviendo.

Dicho proceso comienza evaluando la población y seleccionando los individuos que intervendrán en la formación de la siguiente generación. Seguidamente se aplican los operadores de cruce y mutación y se obtiene la nueva población a partir de la que se repiten los mismos pasos para ir avanzando en el proceso de búsqueda.
A continuación se muestra el seudocódigo de un algoritmo genético simple:

```
Generación de la población inicial
while no se cumpla la condición de parada do
   Evaluación de los individuos
   Selección
   Cruce
   Mutación
end while
```

Cuadro 3.1 Pseudocódico de un AG Simple (Jose Hidalgo, 2002)

3.1.1. Representación genética

Como habíamos expresado antes, las soluciones al problema se codifican mediante cadenas de caracteres denominados cromosomas; Una codificación correcta y eficiente debe representar todo el espacio de soluciones, no debe generar individuos incorrectos al aplicar operadores y debe cubrir todo el espacio de búsqueda de una manera continua. La representación más común suele utilizar un código binario, es decir ceros y unos.

Al conjunto de caracteres que se utilizan para representar la población se le denomina alfabeto y se representa por Ω. Para el caso de la representación binaria sería: Ω = {0,1}. Cada uno de estos valores se denomina alelo. A continuación veremos un ejemplo sencillo de codificación binaria.

Supongamos que tenemos un grupo de 6 personas que trabajan para una empresa. Esta empresa tiene dos oficinas nuevas, una en La Paz y otra en Santa Cruz. Estas 6 personas deben incorporarse a las oficinas en un determinado orden y deben elegir a que oficina van. Una forma sencilla de codificar en binario la distribución de los empleados es utilizar un gen para cada uno de ellos y que el valor de este gen sea 0 si la persona decide irse a La Paz y 1 si elige trabajar en la sede de Santa Cruz.

Así el cromosoma [0 0 0 0 1 1] indicara que las cuatro primeras personas van a La Paz y las 2 últimas trabajarán en Santa Cruz.

En determinados problemas no es suficiente con utilizar una codificación binaria y es necesario hacer uso de otra codificación ya sea con cifras o con caracteres alfanuméricos. Supongamos ahora que esta empresa tiene otros 8 empleados que deben elegir entre 4 destinos; Cochabamba, Oruro, Sucre y Tarija. En este caso las posibles soluciones se pueden representar mediante un cromosoma de 8 genes en el que cada

uno puede tomar un valor del 1 al 4 para indicar la ciudad a la que cada nuevo empleado desea desplazarse.

Si asociamos el alelo 1 a Cochabamba, el 2 a Oruro, el 3 a Sucre y el 4 a Tarija, tendremos el alfabeto: $\Omega = \{1,2,3,4\}$ En este caso el cromosoma [1 2 3 4 4 2 2 3] indicara que la elección de los trabajadores es Cochabamba, Oruro, Sucre, Tarija, Tarija, Oruro, Oruro y Sucre respectivamente.

La codificación binaria es la codificación más extendida, probablemente por razones históricas ya que fue la primera que se usó y es la más aplicable a nuestros sistemas de información binaria. Holland dio una justificación para el uso de codificación binaria. Comparó dos codificaciones que contenían la misma información, una de ellas con un pequeño número de alelos y cadenas largas y otra con número elevado de alelos y cadenas cortas. En el primer caso permita un mayor grado de paralelismo implícito puesto que una instancia del primero contiene más instancias que una del segundo. En contra tiene que la codificación binaria es una codificación no natural y poco útil para muchos problemas. Es bastante usual utilizar representaciones nuevas que permitan evolucionar a elementos más complejos como reglas reconocimiento de patrones o redes neuronales.

3.1.2. Función de coste

Otro aspecto fundamental en el desarrollo de un Algoritmo Genético, es la elección de una buena función de coste. La función de coste debe evaluar a los individuos para indicar cuál es la calidad de la solución que representan y poder realizar el proceso de selección.

Un ejemplo típico es un problema en el cual tenemos una cadena de 6 bits en la que se quiere maximizar el número de unos que contiene la cadena. En este caso la función de coste será $f1(x) = x$, donde x es el numero de 1s que contiene la cadena (individuo)

evaluada. Así el individuo [0 0 0 0 1 1] tendrá asociado un valor de la función de coste igual a 2 y el individuo [0 1 1 0 1 1] un valor de 4.

Normalmente los AG tratan de maximizar una función, pero si lo que se quiere es minimizar, no hay más que utilizar la inversa de la función objetivo, o bien multiplicarla por -1. Para el mismo ejemplo si se quiere minimizar el numero de 1s en la cadena de caracteres, se pueden utilizar las funciones f2(x) = -x ó f3(x) = 1/x.

Volviendo al problema de los 8 empleados del subtítulo anterior, supongamos ahora que el desplazamiento de cada una de las personas tiene un coste para la empresa dependiendo a la ciudad a la que vayan (Tabla 2.1).

Si se quiere minimizar el coste de la nueva distribución no habrá más que utilizar una función de coste de la forma:

$$f(x) = \sum_{i=1}^{8} fi(x)$$

Fórmula 3.1 (Jose Hidalgo, 2002)

Donde fi(x) es el coste de cada empleado para la distribución evaluada. Una de las distribuciones optimas será la representada por el individuo: [2 1 1 4 4 4 2 3]

Empleado	Cochabamba	Oruro	Sucre	Tarija
1	22	13	14	17
2	8	9	10	26
3	7	35	46	12
4	53	28	96	15
5	61	17	38	17
6	42	32	33	15
7	21	12	21	12
8	41	29	27	35

Tabla 3.1: Coste por empleado y desplazamiento para el ejemplo de la función de coste

En muchos casos los AG tienen aparte de una función de coste una función objetivo. Es decir, que aunque la evaluación se realiza de acuerdo a una función se trata de alcanzar un objetivo que se evalúa mediante otra función distinta. Esto suele ocurrir cuando se tratan problemas con restricciones o problemas en los que se pretende optimizar distintos parámetros conocidos como problemas multi-objetivo (Goldberg, 1989). Esta función objetivo no tiene porque ser una función numérica si no simplemente que se cumpla un criterio o no.

3.1.3. Tamaño de la población

Uno de los factores más importante para la convergencia de los algoritmos genéticos es el tamaño de la población. El tiempo necesario para que un AG converja hacia una solución única depende del tamaño de la población. Goldberg y Deb publicaron en 1991 un estudio en el que demuestran que el tiempo para que un individuo se propague a toda la población utilizando los métodos más rápidos de selección es $O(n.\ log_n)$ siendo n el tamaño de la población .

Aunque los AG son eficientes, no garantizan la obtención de una solución óptima. Su efectividad viene claramente determinada por el tamaño de la población. Es muy claro que cuanto mayor sea el número de individuos en la población, se explorarán más zonas del espacio de soluciones; pero también es bastante obvio que esto acarreará un costo computacional mayor. Es por esta razón que se debe buscar una estrecha relación entre el número de individuos utilizados y la calidad que se desea alcanzar.

3.1.4. Operadores de selección

Los algoritmos genéticos necesitan de un operador de selección que identifique a los mejores individuos de la población actual para utilizarlos como padres de la siguiente generación.

Hay muchas formas de realizar la selección, pero siempre se debe asegurar que los mejores individuos tengan una mayor probabilidad de ser seleccionados. Existen varios métodos de selección de los individuos a ser cruzados, a continuación se describe uno de los más utilizados y se mencionan otros que también se consideran importantes:

3.1.4.1. Selección por el método de la ruleta

El método de la ruleta es uno de los más utilizados en la implementación de los AG. La idea es dar a cada individuo una probabilidad de ser seleccionado acorde a su función de coste y proporcional a su "calidad" dentro de la población que se está evaluando. Cuanto mejor es su valor de la función de coste mayor es la probabilidad de ser seleccionado.

Para calcular la probabilidad de selección, *Psi*, de cada individuo *i*, primero se debe evaluar la función de coste para cada uno de ellos, *FFi;* y a continuación se calcula la suma de todas ellas y se obtiene Psi de acuerdo a la siguiente expresión:

$$Psi = \frac{FFi}{\sum_{i=1}^{n} FFi}$$

Donde *n* es el número de individuos de la población.

Fórmula 3.2 (Jose Hidalgo, 2002)

A continuación se calcula la probabilidad de selección acumulada *Pai* para cada individuo. Para calcularla se van sumando las probabilidades de selección de los individuos:

$$Pai = \sum_{j=0}^{n} Psj$$

Fórmula 3.3 (Jose Hidalgo, 2002)

Luego se generan tantos números aleatorios como individuos queramos seleccionar. Cada número aleatorio se compara con las probabilidades acumuladas y se escoge el

individuo que tenga asociada una probabilidad inmediatamente menor al número aleatorio.

Continuando con nuestro ejemplo supongamos la población de 4 individuos con sus correspondientes valores de la función de coste *Ffi*, junto con las probabilidades *Psi* y *Pai*, que se muestra en la siguiente tabla:

N	individuo	*FFi*	*Psi*	*Pai*
1	11223344	211	0.209	0.209
2	44231221	320	0.317	0.526
3	12341234	241	0.239	0.765
4	22232421	238	0.235	1.000

Tabla 3.2: Ejemplo de selección por el método de la ruleta.

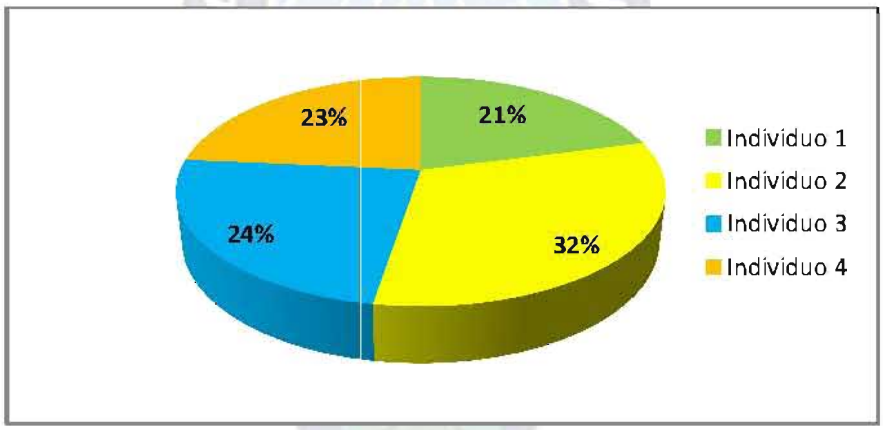


Figura 3.1 probabilidad porcentual por individuo

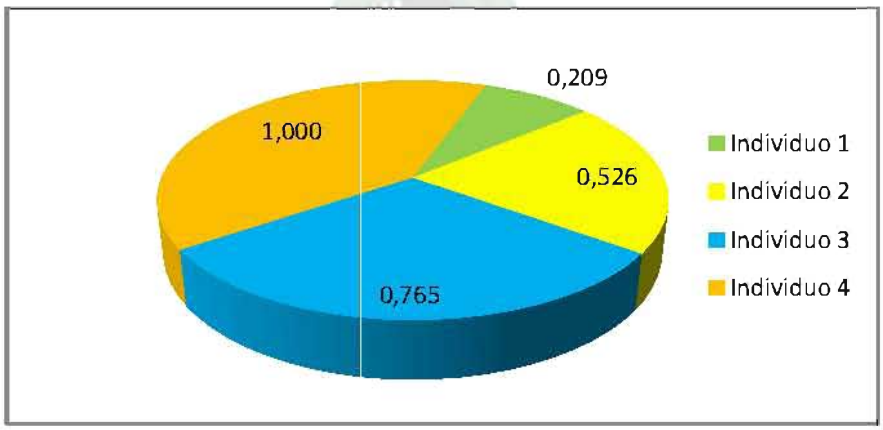


Figura 3.2 Probabilidad acumulada por individuo

La siguiente tabla muestra los individuos que serán seleccionados para 4 números aleatorios:

Nº aleatorio	N	Selección
0.23	1	11223344
0.07	4	44231221
0.97	3	12341234
0.65	2	22232421

Figura 3.3 Selección de individuos

3.1.4.2. Otros métodos de Selección

* Elitismo: Consiste en guardar siempre el mejor individuo de la población para la siguiente generación, generalmente sustituyéndolo por el peor. Hay estudios que indican que un algoritmo con selección elitista asegura la convergencia del AG hacia un óptimo global.

* Selección Sigma "Forrest": Es una técnica usada precisamente para intentar adaptar la selección según evoluciona el algoritmo. Con esta técnica el valor esperado de un individuo depende de su fitness, del fitness medio de la población y de la desviación estándar de la población.

* Selección basada en el rango: En este tipo de selección se mantiene un porcentaje de la población para la siguiente generación. La población se ordena por orden de fitness, y los K peores se eliminan y se sustituyen por descendientes de los K mejores con algún otro individuo de la población.

* Selección de Boltzman: Funciona de manera similar a como funciona el enfriamiento simulado, variando la temperatura que controla la presión de selección. La temperatura inicial debe ser muy elevada lo que significa que la presión de selección es baja y por

lo tanto se prima la exploración. La temperatura se va bajando gradualmente con lo que incrementa la presión de selección.

* Selección por torneo: Esta técnica es similar a la de rango en lo que a medida de presión se refiere, pero es eficiente computacionalmente hablando y se puede paralelizar más cómodamente. Se escogen dos individuos aleatoriamente de la población, se genera un numero aleatorio r comprendido entre 0 y 1. Si r < k, donde k es un parámetro, se selecciona el mejor de los dos individuos en caso contrario se selecciona el peor. Los dos se devuelven a la población inicial para que puedan ser seleccionados de nuevo.

3.1.5. Operadores de cruce

La operación de cruce consiste en escoger aleatoriamente un par de individuos de los seleccionados e intercambiar una parte de sus cromosomas entre sí; además este operador se aplica con una determinada probabilidad.

El intercambio del código se realiza a partir de unos puntos que se seleccionan también aleatoriamente, la forma más común de realizar el cruce es seleccionar un único punto e intercambiar el código a partir del mismo; comúnmente llamado cruce por un punto. Otro tipo de cruce es el cruce uniforme, seleccionando un patrón de intercambio. Por supuesto también existen otros mecanismos de cruce.

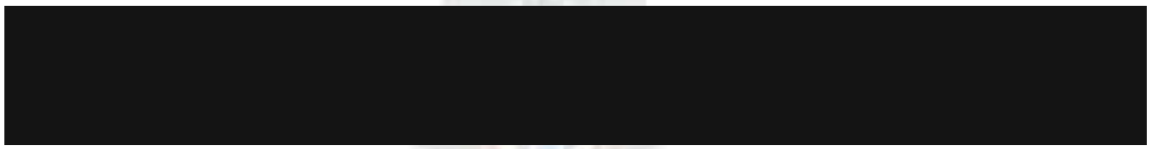


Figura 3.3: Operador de cruce

Un ejemplo de esta operación de cruce, se muestra a partir de los dos primeros individuos de la tabla 3.2, tomando como punto de cruce el gen número 3:

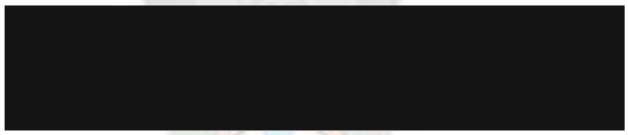
Para resolver el problema de las soluciones falsas existen varias soluciones como la doble codificación o mecanismos reparadores.

3.1.6. Operadores de mutación

El operador de mutación es aquel que está encargado de realizar pequeños cambios en el código con una probabilidad muy pequeña, se utiliza para restablecer la diversidad que se haya podido perder con la aplicación sucesiva de los operadores de selección y cruce.

Ochoa ha realizado un estudio en el que demuestra la importancia de las proporciones de mutación y de selección en la eficiencia del algoritmo.

También existen otros trabajos en los que se utilizan valores de las probabilidades de mutación que varían con la progresión del algoritmo. El método más habitual de realizar la mutación es seleccionar un gen del individuo y cambiar su alelo por otro de los del alfabeto. Un ejemplo de una mutación en el 3er gen es:



La probabilidad con la que se da las mutaciones suele ser baja, para evitar que el AG realice una búsqueda aleatoria. Sin embargo en ocasiones puede ser necesario aumentar esta probabilidad para recuperar la diversidad de la población. En (Goldberg, 1989) se menciona el diseño de un nuevo operador genético, el operador de Regeneración para evitar problemas de convergencia anticipada, que se basa en esta afirmación y que ha dado buenos resultados.

3.2. ALGORITMOS GENÉTICOS PARALELOS

Se considera a un programa paralelo, si en cualquier momento de su ejecución puede ejecutar más de un proceso. Para crear programas paralelos eficientes hay que poder crear, destruir y especificar procesos así como la interacción entre ellos. Básicamente existen tres formas de paralelizar un programa:

- **Paralelización de grano fino**: la paralelización del programa se realiza a nivel de instrucción; cada procesador hace una parte de cada paso del algoritmo (selección, cruce y mutación) sobre la población común.
- **Paralelización de grano medio**: los programas se paralelizan a nivel de bucle. Esta paralelización se realiza habitualmente de una forma automática en los compiladores.
- **Paralelización de grano grueso**: se basan en la descomposición del dominio de datos entre los procesadores, siendo cada uno de ellos el responsable de realizar los cálculos sobre sus datos locales.

La paralelización de grano grueso tiene como gran atractivo la portabilidad, ya que se adapta perfectamente tanto a multiprocesadores de memoria distribuida como de memoria compartida. Este tipo de paralelización a su vez, se puede realizar siguiendo tres estilos distintos de programación: paralelismo en datos, programación por paso de mensajes y programación por paso de datos.

- **Paralelismo en datos:** El compilador se encarga de la distribución de los datos guiado por un conjunto de directivas que introduce el programador. Estas directivas hacen que cuando se compila el programa las funciones se distribuyan entre los procesadores disponibles. Como principal ventaja presenta su facilidad de programación. Los lenguajes de paralelismo de datos más utilizados son el estándar HPF "High Performance Fortran" y el OpenMP.

- **Programación por paso de mensajes**: El método más utilizado para programar sistemas de memoria distribuida es el paso de mensajes o alguna variante del mismo. La forma más básica consiste en que los procesos coordinan sus actividades mediante el envío y la recepción de mensajes. Las librerías más utilizadas son por este orden la estándar MPI "Message Passing Interface" y ***PVM "Parallel Virtual Machine"***.

- **Programación por paso de datos:** A diferencia del modelo de paso de mensajes, la transferencia de datos entre los procesadores se realiza con primitivas unilaterales tipo put-get, lo que evita la necesidad de sincronización entre los procesadores emisor y receptor. Es un modelo de programación de muy bajo nivel pero muy eficiente, aunque en la actualidad son muy pocos los fabricantes que los soportan.

La computación paralela se ha convertido en una parte fundamental en todas las áreas de cálculo científico, ya que permite la mejora del rendimiento simplemente con la utilización de un mayor número de procesadores, memorias y la inclusión de elementos de comunicación que permitan a los procesadores trabajar conjuntamente para resolver un determinado problema.

Para el caso que estamos estudiando, podemos decir que los algoritmos genéticos tienen una estructura que se adapta perfectamente a la paralelización, de hecho la evolución natural es en sí un proceso paralelo ya que evoluciona utilizando varios individuos en diferentes áreas y circunstancias. Los principales métodos de paralelización de un AG consisten en la división de la población en varias subpoblaciones llamadas demes; el tamaño y distribución de la población entre los distintos procesadores es uno de los factores fundamentales a la hora de paralelizar el algoritmo.

3.2.1. Clasificación de los Algoritmos Genéticos Paralelos

Existen diversas formas de paralelizar un algoritmo genético. La primera y más intuitiva es la global, que consiste básicamente en paralelizar la evaluación de los individuos

manteniendo una población; otro método de paralelización global consiste en realizar una ejecución de distintos AG secuenciales simultáneamente. El resto de las aproximaciones dividen la población en subpoblaciones que evolucionan de forma independiente e intercambian individuos cada cierto número de generaciones, si las poblaciones son pocas y grandes, tenemos la paralelización de grano grueso, si el número de poblaciones es grande y con pocos individuos en cada población tenemos la paralelización de grano fino.

Finalmente, existen algoritmos que combinan propiedades de estos dos últimos, los cuales son denominados mixtos. En la siguiente figura podemos ver un esquema de la clasificación de los Algoritmos Genéticos Paralelos AGP.

Figura 3.4: Clasificación de los AGP (Jose Hidalgo, 2002)

Además de obtener tiempos de procesamiento menores, al paralelizar un algoritmo genético, estamos modificando el comportamiento algorítmico, y esto hace que logremos obtener otras soluciones y experimentar con las distintas posibilidades de implementación y los distintos factores que influyen en ella.

Estas poblaciones van evolucionando por separado para detenerse en un momento determinado e intercambiar los mejores individuos entre ellas.

Técnicamente hay 3 características importantes que influyen en la eficiencia de un algoritmo genético paralelo:

- La topología que define la comunicación entre subpoblaciones.
- La proporción de intercambio: número de individuos a intercambiar
- Los intervalos de migración: periodicidad con que se intercambian los individuos.

A continuación se muestra el seudocódigo de un algoritmo genético con la evaluación en paralelo y que mantiene la misma población durante todo el proceso de cálculo:

```
generación de la población inicial
while no se cumpla la condición de parada do
  do in parallel
       evaluación de los individuos
  end parallel do
       selección
       cruce
       mutación
end while
```

Cuadro 3.2 Pseudocódico de un AG con evaluación en paralelo (Jose Hidalgo, 2002)

En el siguiente esquema se muestra el seudocódigo de un AG con varias poblaciones que evolucionan en paralelo (aplica tanto para grano fino como para grano grueso), adicionalmente si se desea, se podría especificar la metodología de migración para los individuos seleccionados entre las poblaciones:

```
do in parallel
  generación de la población inicial
  while no se cumpla la condición de parada do
        evaluación de los individuos
        selección
        producción de nuevos individuos
        mutación
  end while
end parallel do
Escoger la mejor solución
```

Cuadro 3.3 Pseudocódico de un AG con poblaciones que evolucionan en paralelo (Jose Hidalgo, 2002)

CAPITULO 4

DESARROLLO DE UNA TÉCNICA EVOLUTIVA APLICADA A LA RESOLUCIÓN DEL PROBLEMA DE STEINER

4. DESARROLLO DE UNA TECNICA EVOLUTIVA APLICADA A LA RESOLUCIÓN DEL PROBLEMA DE STEINER

RESUMEN

En este capítulo se describe el algoritmo genético propuesto como solución al problema de Steiner para el diseño de redes de comunicación confiables. Se describe la codificación, los operadores, la función fitness a considerar y además el cálculo de factibilidad. Se describe también la implementación del AG y finalmente se muestran los casos de prueba junto a los resultados obtenidos.

El objetivo principal de los señalados en el presente trabajo, es investigar la aplicabilidad de los algoritmos genéticos para resolver problemas de confiabilidad en una red de comunicación de computadoras, tomando en cuenta la dimensión de este problema se ha propuesto implementar un modelo evolutivo para el manejo eficiente de la población de posibles soluciones.

No existen antecedentes sobre la resolución del problema generalizado de Steiner mediante técnicas evolutivas fuera de la mencionada en (Nesmachnow, 2002). Existen trabajos sobre AG aplicados al problema del árbol de mínima expansión de Steiner, pero fuera de nuestro entorno de trabajo no hay referencias de aplicación de técnicas evolutivas al problema generalizado.

El algoritmo genético diseñado en este contexto, corresponde a un modelo simple. En este capítulo se presenta los detalles de codificación e implementación del AG utilizado para la resolución del GSP. Los operadores de cruzamiento, mutación, además de los procesos de selección y cálculo de factibilidad constituyen la parte fundamental de nuestro estudio y son presentados en detalle más adelante, también describimos algunas consideraciones a tener cuando implementamos el AG.

4.1. CONSIDERACIONES

Los algoritmos genéticos han probado su eficacia en caso de querer calcular funciones no derivables o de derivación muy compleja, aunque su uso es posible con cualquier función. Deben de tenerse en cuenta también algunas consideraciones importantes:

- Si la función a optimizar tiene muchos máximos/mínimos locales se requerirán más iteraciones del algoritmo para "asegurar" el máximo/mínimo global.
- Si la función a optimizar contiene varios puntos muy cercanos en valor al óptimo, solamente podemos asegurar que encontraremos uno de ellos y no necesariamente el óptimo.

Como cualquier otro método, el uso de un Algoritmo Genético para resolver un problema, debe tener un análisis previo, por lo que resulta muy importante usar el conocimiento que se tiene del sistema que se desea optimizar para determinar si el algoritmo genético tendrá un desempeño adecuado.

Es importante considerar también que aún para algunos casos donde se sugiere que otras técnicas trabajarán mejor que los AG, con suficiente experimentos, se podrá obtener buen desempeño en la búsqueda usando los mismos AG.

Por supuesto, tomar cualquier método de optimización para trabajar de una manera óptima requiere alguna experimentación con sus parámetros de configuración. Selecciones inadecuadas de estos parámetros conducirán a un bajo rendimiento en la búsqueda.

Por el contrario, para los problemas donde el cálculo de las soluciones es numéricamente preciso y rápido, no se recomienda usar los AG, ya que éstos alcanzarán la región óptima mucho más lento que los métodos determinísticos.

Otro tipo de aplicaciones no recomendadas para los AG son aquellas que requieren encontrar el óptimo global exacto, dada la característica de los AG de ser buenos encontrando la región óptima global pero enfrentando problemas algunas veces para localizar el óptimo exacto.

Tomando en cuenta las consideraciones citadas, queda ahora presentar el modelo del AG propuesto.

4.2. MODELO DEL ALGORITMO GENÉTICO

Un AG puede presentar diversas variaciones, esto dependiendo de como se aplican los operadores genéticos de cruzamiento, mutación y/o algún otro, depende también de como se realiza la selección de los individuos a reproducirse y de como se decide el reemplazo de los individuos para formar la nueva población.

A continuación se presenta los detalles de codificación, operadores y funciones del AG utilizado en la propuesta para la resolución del GSP.

4.2.1. Esquema general

El pseudocódigo del AG propuesto es el siguiente:

```
Inicializar(P(0))
generacion = 0
Evaluar(P(0))
mientras (no CriterioParada) hacer
     Padres = Seleccion(P(generacion))
     Hijos = Operadores de Reproduccion(Padres)
     NuevaPop = Reemplazar(Hijos,P(generacion))
     generacion ++
     P(generacion) = NuevaPop
retornar Mejor Solucion Hallada
```

Cuadro 4.1 Pseudocódigo del AG propuesto, adaptación de (Nesmachnow, 2002)

Describiendo el esquema anterior. En primera instancia se genera la población inicial de individuos, los cuales son grafos probabilísticos representados genéticamente y que son posibles soluciones al GSP, esta población es evaluada antes de comenzar la evolución para descartar que a priori se tengan las soluciones optimas.

El proceso de evolución se inicia y continúa mientras no se cumpla el criterio de parada, el cual es una combinación de un número máximo de generaciones y la búsqueda de un individuo mejor adaptado. Un individuo mejor adaptado será aquel que tenga un mejor valor de fitness, el cual está dado por el costo total de los enlaces del grafo que representan a la red de Steiner, pero que además satisfaga los criterios de conexión del GSP

Durante la evolución se sigue el proceso natural de selección y que además es el más general en los AG. Se selecciona a los padres y se realiza el cruzamiento, el cual da como resultado dos nuevos grafos y que nuevamente son posibles soluciones al GSP.

En este ciclo también se introduce el operador de mutación que ayuda a mejorar el espacio de búsqueda y ayuda a prevenir una convergencia prematura del algoritmo. Se hace la evaluación de los nuevos grafos para evaluar su fitness y se ingresa en la nueva generación a los individuos mejor adaptados.

Una vez se cumpla la condición de parada o el algoritmo haya convergido, se devuelve al individuo mejor adaptado. Este debe cumplir todos los requisitos de conexión prefijados en el problema y tener el menor costo posible.

La solución está representada por el siguiente esquema:

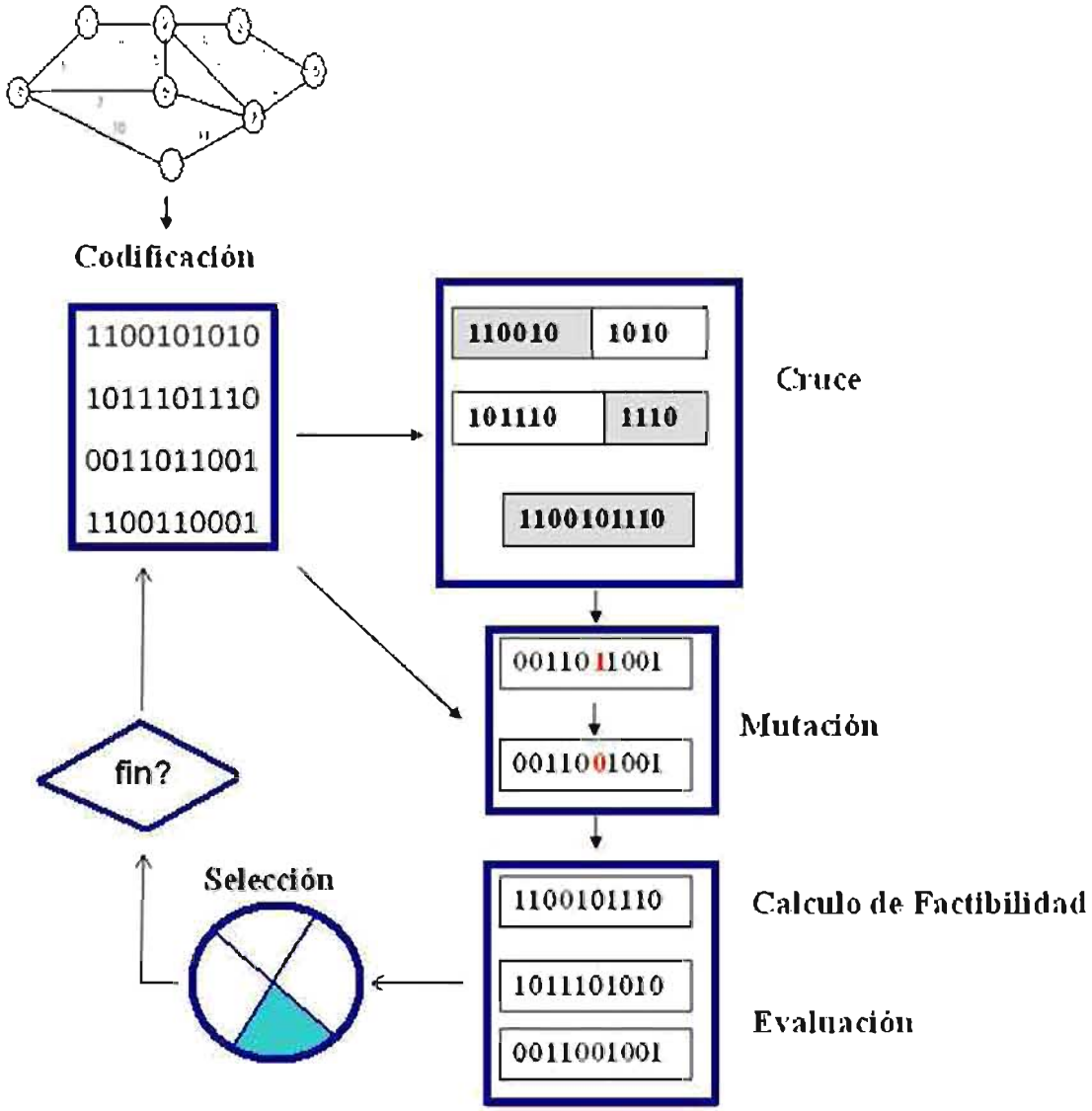


Figura 4.1 Esquema de funcionamiento del AG, según (Jose Hidalgo, 2002)

4.2.2. Codificación

El algoritmo genético a implementar se basa en una codificación binaria sencilla. Cada cromosoma consta de un arreglo de bits que representan a las aristas del grafo original, numeradas de 0 a número de aristas - 1.

La presencia o ausencia de una arista en un grafo solución queda determinada por el valor 1 o 0 respectivamente, en la posición correspondiente en el cromosoma.

La Figura 4.1 presenta un ejemplo de la codificación binaria utilizada para un grafo pequeño, donde los nodos terminales se han coloreado en oscuro y los nodos de Steiner en claro.

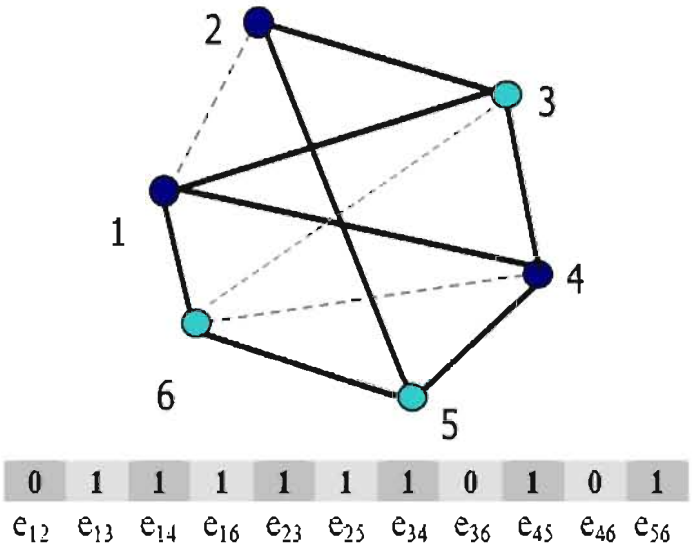


Figura 4.1 Ejemplo de codificación binaria para el AG (Sergio Nesmachnow, 2002)

Las aristas marcadas con línea solida son aquellas presentes en el grafo representado y se notan con el valor 1 en el cromosoma de bits, las aristas marcadas con la línea punteada gris están presentes en el grafo original pero no en el grafo solución representado, por lo cual se codifican con el valor 0 en el cromosoma. Las etiquetas de las posiciones del cromosoma referencian a los nodos origen y destino de cada arista del grafo original por ejemplo: "e25 = arista entre nodo 2 y nodo 5".

Al utilizar la representación binaria los operadores de cruzamiento o mutación resultan sencillos, pero tienen el inconveniente de que los cromosomas resultantes, descendientes y mutaciones, pueden no representar a una solución factible del GSP.

Por tal motivo, es necesario verificar la factibilidad de los nuevos cromosomas creados al aplicar un operador evolutivo.

En la implementación del algoritmo se ha adoptado el criterio de descartar individuos no factibles. Esta decisión evita la tarea de cuantificar cuán lejos de una solución factible se encuentra un individuo no factible y determinar un valor adecuado de penalización para su fitness. Esta idea sigue la propuesta de (Esbensen, 1995) en su trabajo sobre el problema del árbol de Steiner.

4.2.3. Inicialización

La población inicial se genera aleatoriamente, la misma está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema.

En caso de que no se lo quiera hacer de forma aleatoria, es importante garantizar que dentro de la población inicial, se tenga la diversidad estructural de estas soluciones para tener una representación de la mayor parte de la población posible o al menos evitar la convergencia prematura.

4.2.4. Selección

Se ha elegido una variación del método elitista, el cual elige a los individuos más fuertes o más aptos para sobrevivir y descarta a aquellos que son menos aptos o más débiles. Este método por lo general acelera la convergencia pero también reduce el campo de búsqueda por lo se debe poner especial atención en el resto de los operadores que intervienen en el AG.

Por consiguiente se eligen a los individuos que mejor función de costo representan, es decir, cuyo cálculo de fitness sea mejor o más alto.

4.2.5. Cruzamiento

El operador de cruce es altamente responsable de las propiedades del algoritmo genético y determinará en gran medida la evolución de la población.

Para nuestro caso en particular se ha tomado una variación de la técnica de un punto, donde se escoge de forma aleatoria la posición de los alelos heredados, es decir que los alelos del padre pueden estar primero o después a los de la madre, esto amplia el espacio de búsqueda que queda reducido con el método de selección que se ha escogido.

4.2.6. Mutación

La mutación es una variación de las informaciones en el código genético. En la presente propuesta es un factor determinante por lo que se ha elegido uno de los métodos más agresivos.

El método, es la mutación multibit donde cada alelo tiene una probabilidad de mutarse o no y que es calculada en cada pasada para cada individuo después del cruce. Esto también va en el sentido de ampliar el espacio de búsqueda.

4.2.7. Función de Fitness

La función de fitness utilizada en la propuesta, toma en cuenta el costo del grafo representado por un cromosoma. Sustrayendo el valor de costo de un individuo del valor del costo del grafo original es posible mapear el problema de minimización del costo del grafo en el problema de maximización de la función presentada en la siguiente ecuación.

$$f = C_{ORIG} - \sum_{0}^{|E|-1} [EDGE(i) * C(i)]$$

Fórmula 4.1 (Nesmachnow)

En la Ecuación, |E| denota la cardinalidad del conjunto de aristas del grafo original, correspondiente al largo de la codificación utilizada. C_{ORIG} representa el costo del grafo original.

La función C: [0,|E|-1] → R devuelve el costo de una arista y la función EDGE: [0,|E|-1] → [0,1] retorna el valor binario correspondiente a la arista que ocupa la posición i-ésima en el cromosoma (Nesmachnow).

4.2.8. Condición de Término

Se ha trabajado con un criterio de parada combinado, en el cual se considera los dos criterios más conocidos:

- Por un lado, el proceso termina si no hay cambios en la población que mejoren la función de costo después de un número generaciones.
- En segundo lugar, concluye después de un esfuerzo prefijado en un número mayor de generaciones, donde no se encuentran soluciones factibles.

El primero de alguno de los criterios que se cumpla detiene el proceso y entrega la mejor solución encontrada.

4.2.9. Cálculo de factibilidad

La decisión de trabajar con individuos factibles simplifica el diseño de los operadores evolutivos, pero implica utilizar un procedimiento adicional para determinar la factibilidad de individuos luego de aplicar un operador. El cálculo de factibilidad verifica que el grafo representado cumpla las restricciones del GSP.

El algoritmo para cálculo de factibilidad consta de dos etapas:

Una heurística simple se utiliza para detectar soluciones no factibles, verificando que los grados de los nodos terminales sean mayores o iguales al número máximo de caminos impuestos como restricción para cada nodo.
La complejidad de esta heurística es de orden cuadrático en la cardinalidad del conjunto de nodos terminales y evita el cálculo de caminos entre pares de nodos terminales en todos los casos (Nesmachnow, 2002).

Que un grafo pase el chequeo heurístico de grados de nodos terminales no implica que sea solución factible del GSP. En tal caso deben hallarse los caminos entre cada par de nodos terminales. Se ha utilizado una variante del algoritmo de Ford–Fulkerson para cálculo de flujo máximo, que permite hallar caminos entre pares de nodos terminales asignando a un nodo el rol de fuente y al otro el rol de pozo.

El algoritmo tradicional de Ford–Fulkerson trabaja sobre grafos dirigidos, por lo cual en el caso del grafo del GSP cada arista del grafo se considera como un par de aristas de sentidos opuestos. Asumiendo capacidad unitaria para las aristas, el flujo máximo entre fuente y pozo coincide con el número máximo de caminos disjuntos entre ellos. Si éste es menor que el valor del requerimiento correspondiente, el grafo no es factible.

La variante de Ford–Fulkerson descrita utiliza una función de capacidad de flujo unitaria para cada arista, y tiene orden $O(|E|\ r_{MAX}\ n_t^2)$, siendo $|E|$ la cardinalidad del conjunto de aristas del grafo original, r_{MAX} el valor máximo del conjunto de requerimientos, y n_t la cardinalidad del conjunto de nodos terminales (Nesmachnow).

4.3. IMPLEMENTACION DEL ALGORITMO GENÉTICO

En este punto se explica la implementación del algoritmo genético, se brindan detalles sobre los parámetros de configuración del algoritmo, se señala la plataforma sobre la cual se han hecho las pruebas y una validación de los resultados frente a otros ya conocidos.

4.3.1. Configuración de parámetros

Los operadores utilizados por el AG en función de un análisis previo de resultados se configuraron de la siguiente manera:

a) Selección Elitista de 2 individuos por generación.

b) Cruzamiento de composición aleatoria con probabilidad del 50% para padre y madre de 1 punto fijo.

c) Mutación multibit, para cada alelo del cromosoma optimo se ha establecido una probabilidad de mutación de 10%; elevada en comparación a los parámetros tradicionales q varían entre el 3% y 7%).

d) Función de fitness, tal cual la descrita en párrafos anteriores; como la suma de costos generados aleatoriamente para cada ejemplo.

e) Cálculo de factibilidad adicional; grado y flujo máximo superiores o iguales al requerimiento de enlaces redundantes.

4.3.2. Plataforma de Ejecución

Para codificar el algoritmo genético se utilizó Visual Basic 6.0 en su forma nativa, ya que todos los módulos y operadores fueron desarrollados según las necesidades específicas.

Para las pruebas de validación del algoritmo, se utilizo un equipo similar al descrito en las pruebas de (Robledo, 2000) , esto con el fin de comparar de manera equitativa las soluciones.

El equipo tiene las siguientes características:

- Procesador Pentium 3 de 600 MHz
- Memoria RAM de 128 MB
- Sistema Operativo Windows 2000 estándar

4.3.3. Validación

Para validar el funcionamiento del AG propuesto, se ha utilizado los resultados conocidos de una versión serial que resuelve el GSP y que esta descrita en (Robledo, 2000), los grafos de prueba han sido extraídos de la misma fuente, posibilitando la comparación de los resultados obtenidos.

La condición de parada ha sido establecida en 100 generaciones sin cambios en la función de coste óptima y 1000 generaciones sin encontrar nuevas soluciones factibles.

Las características y comparaciones de los grafos utilizados se presentan en la siguiente tabla:

Tabla 4.1 comparación de resultados entre el algoritmo genético propuesto y los resultados conocidos en (Robledo, 2000)

Los resultados muestran una dispersión entre resultados iguales, mejores y peores. La validación se realizo asumiendo costo unitario en todos los enlaces, por lo que se pueden esperar mejores resultados con el AG.

4.3.4. Ventajas

En base a la descripción del modelo planteado, se espera que el AG pueda obtener soluciones factibles, óptimas y en tiempos razonables.

Las ventajas del modelo planteado son:

- Se deduce una rápida convergencia del algoritmo, sin reducir demasiado el espacio de exploración.
- La representación es sencilla y facilita la aplicación de los operadores de cruzamiento y mutación.
- Se mejora los tiempos al descartar soluciones no factibles.
- Al introducir conocimiento específico del problema se sugiere la existencia de soluciones más cercanas al óptimo global y en algunos casos se espera que se llegue al óptimo.

CAPITULO 5

VALIDACIÓN DEL MODELO

5. VALIDACION DEL MODELO

Tal cual se ha descrito en los capítulos anteriores, una red puede ser representada mediante un grafo simple G = {V, E, C}, donde V es el conjunto de vértices o nodos, E el conjunto de enlaces o aristas y C el conjunto de costos asociados a cada arista que pertenece a E.

El mismo grafo a su vez puede ser representado mediante matrices, con el objetivo de denotar y demostrar su funcionamiento. A continuación veremos en detalle esta representación.

5.1. REPRESENTACIÓN MATRICIAL DE GRAFOS

Los grafos se constituyen en una herramienta útil para el estudio y modelado topológico de redes de todo ámbito.

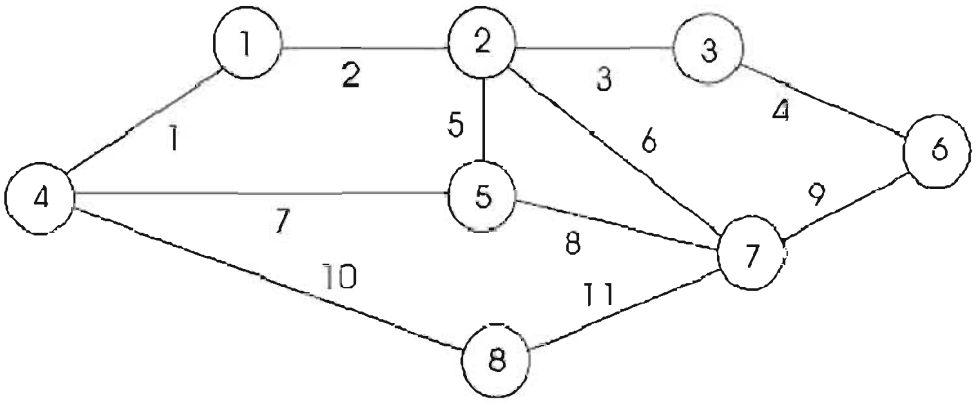


Figura 5.1: Grafo simple en representación de una red (Magnago, 2006)

Para poder analizarlos y que puedan ser procesados computacionalmente es necesario representarlos de forma matricial, por consiguiente ahora veremos cómo se hace esta representación (Magnago, 2006):

5.1.1. Matriz incidente

Sea un grafo **G** con *n* nodos y *m* enlaces, sin lazos. Se define la matriz **A** de dimensión *n* por m tal que $\mathbf{A}=[\mathbf{a_{ij}}]$, donde *n* filas corresponden a *n* nodos y las *m* columnas corresponden a *m* enlaces. Por ende $a_{ij} = 1$ si el j-ésimo enlace e_i es incidente al i-ésimo nodo v_i ; será igual a 0 en otro caso.

Esta matriz **A** se denomina incidencia nodo-enlace o simplemente matriz de incidencia. A veces, si la matriz **A** representa al grafo **G**, se simboliza como **A(G).**

En la figura siguiente se muestra la representación en una matriz incidente para el grafo de la figura 5.1.

$$A(G) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Figura 5.2 Matriz de incidencia para el grafo de la figura 5.1 (Magnago, 2006)

En esta representación matricial se puede observar que dado que los enlaces son incidentes en dos nodos, cada columna de A tiene exactamente dos unos. La cantidad de unos en cada fila representa la cantidad de enlaces que inciden sobre el nodo asociado la cual se define como grado de un nodo. Una fila con todos 0, representa un nodo aislado y finalmente, la permutación de dos filas o columnas cualesquiera de A(G), simplemente resulta en una renumeración de los nodos o enlaces del mismo grafo G.

5.1.2. Matriz de adyacencia y matriz de costo

Sean n el número de nodos y m la cantidad de enlaces de una red de datos representada por un grafo simple G. Luego, la matriz de conexión M[m_{ij}] es una matriz binaria de dimensión n por n y donde:

$$Mij = \begin{cases} 1 \textit{ si el enlace que une los nodos i y j está disponible} \\ 0 \textit{ en caso contrario} \end{cases}$$

La siguiente figura muestra la matriz de adyacencia del grafo representado en la figura 5.1

$$\mathbf{C}(G) = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 5.3 Matriz de adyacencia para el grafo de la figura 5.1 (Magnago, 2006)

La matriz de costo C[c_{ij}] tiene la misma dimensión que la matriz de adyacencia donde cada elemento *cij* representa el costo del enlace entre los nodos i y j. Así, mientras que la matriz de adyacencia representa un grafo simple, la matriz de costo representa un grafo simple ponderado.

Los ceros se mantienen invariantes, asumiendo que un costo nulo representa un enlace que no está disponible, esto tiene mayor coherencia matemática, que si un enlace inexistente se representara por ∞, pero habitualmente se utiliza la representación con 0 dado que permite al algoritmo interpretar redes no totalmente conectadas.

La siguiente figura muestra la matriz de costo asociada al grafo representado en la figura 5.1, cuyos valores han sido elegidos aleatoriamente.

$$\mathbf{C}(G) = \begin{bmatrix} 0 & 18 & 0 & 21 & 0 & 0 & 0 & 0 \\ 18 & 0 & 32 & 0 & 10 & 0 & 12 & 0 \\ 0 & 32 & 0 & 0 & 0 & 17 & 0 & 0 \\ 21 & 0 & 0 & 0 & 11 & 0 & 0 & 18 \\ 0 & 10 & 0 & 11 & 0 & 0 & 18 & 0 \\ 0 & 0 & 17 & 0 & 0 & 0 & 15 & 0 \\ 0 & 12 & 0 & 0 & 18 & 15 & 0 & 13 \\ 0 & 0 & 0 & 18 & 0 & 0 & 13 & 0 \end{bmatrix}$$

Figura 5.4 Matriz de costos para el grafo de la figura 5.1 (Magnago, 2006)

Las matrices M(G) y C(G), son siempre cuadradas, dado que tanto las filas como las columnas representan el mismo parámetro y son simétricas por que el enlace que une el i-ésimo con el j-ésimo nodo es el mismo que une el j-ésimo con el i-ésimo nodo. Al no estar permitidos los lazos, la diagonal es siempre nula.

Para el presente estudio y tomando en cuenta que toda red puede ser representada mediante una matriz de adyacencia asociada a una matriz de costos, podemos analizar ciertos casos de prueba que utilizan esta representación matricial.

A la vez estas matrices son transformadas a su correspondiente codificación genética para su evaluación mediante el método propuesto.

5.2. CASOS DE PRUEBA

Por los pocos estudios encontrados sobre la aplicación de técnicas evolutivas aplicadas a la resolución del GSP, no existen conjuntos de prueba estándares.

Para poder probar el algoritmo se ha diseñado tres casos donde se ha seleccionado aleatoriamente tanto las topologías de conexión como los costos de cada enlace, el número de requerimientos se escoge de manera aleatoria con valores de entre 0 y 3.

La siguiente tabla resume las características de los grafos de prueba, detallando el número total de nodos, nodos terminales, aristas y sus requerimientos.

Tabla 5.1 Detalles de los grafos de prueba

Su representación matricial como su modelo de grafo, son mostrados en los resultados de cada caso para su mejor comprensión. Los costos de los enlaces son generados aleatoriamente al momento de inicializar el grafo original.

Los grafos originales más los resultados obtenidos y la explicación de los mismos se muestran a continuación:

5.2.1. Resultados

5.2.1.1. Caso 1

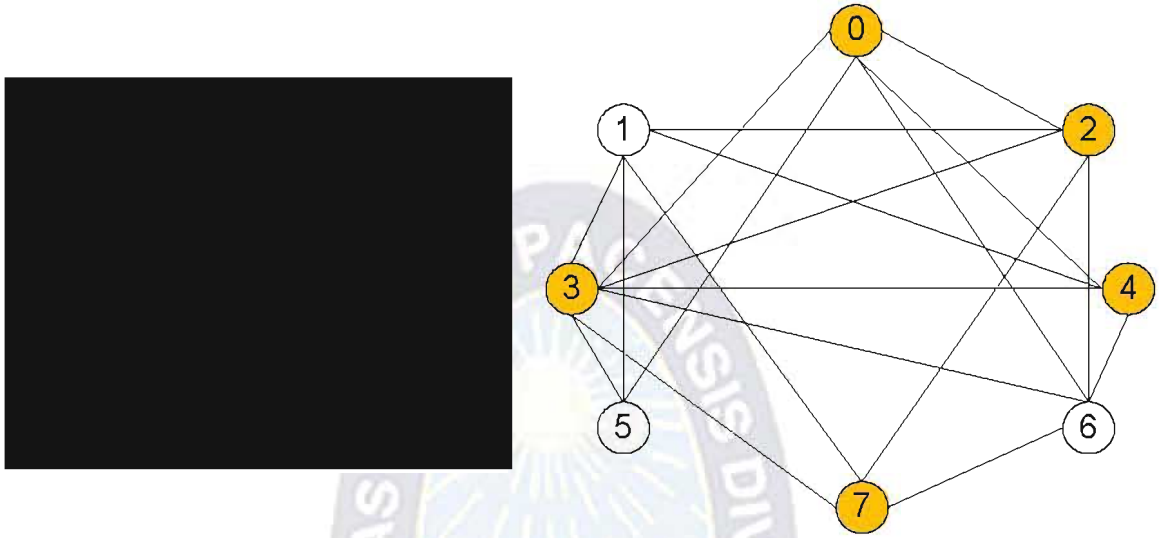


Figura 5.5 Grafo original para el caso de prueba 1

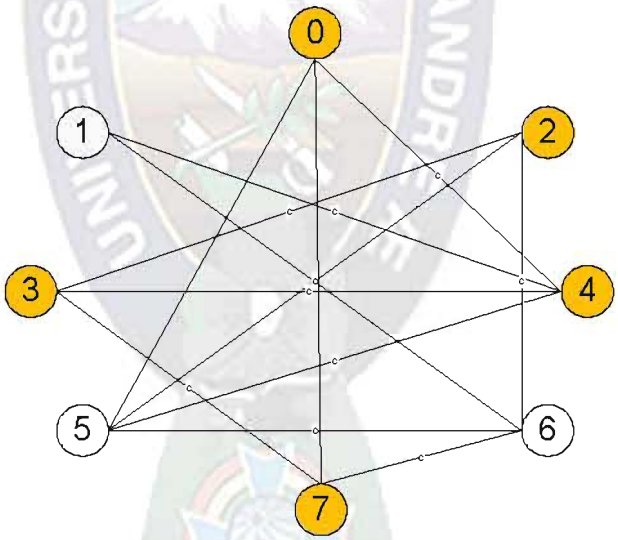


Figura 5.6 Grafo optimizado para el caso de prueba 1

Tabla 5.2 Resultado de la prueba sobre el caso 1

Los resultados que se observan en la Tabla 5.2 muestran la optimización hecha a partir del grafo original respecto al grafo resultante. Hubo una reducción en el número de enlaces, estos disminuyeron de 21 a 13, mientras que el costo total de la red se redujo de $us. 43.31 a $us 16.63, el tiempo que le tomo al algoritmo encontrar estos resultados fue de 0.33 segundos.

5.2.1.2. Caso 2

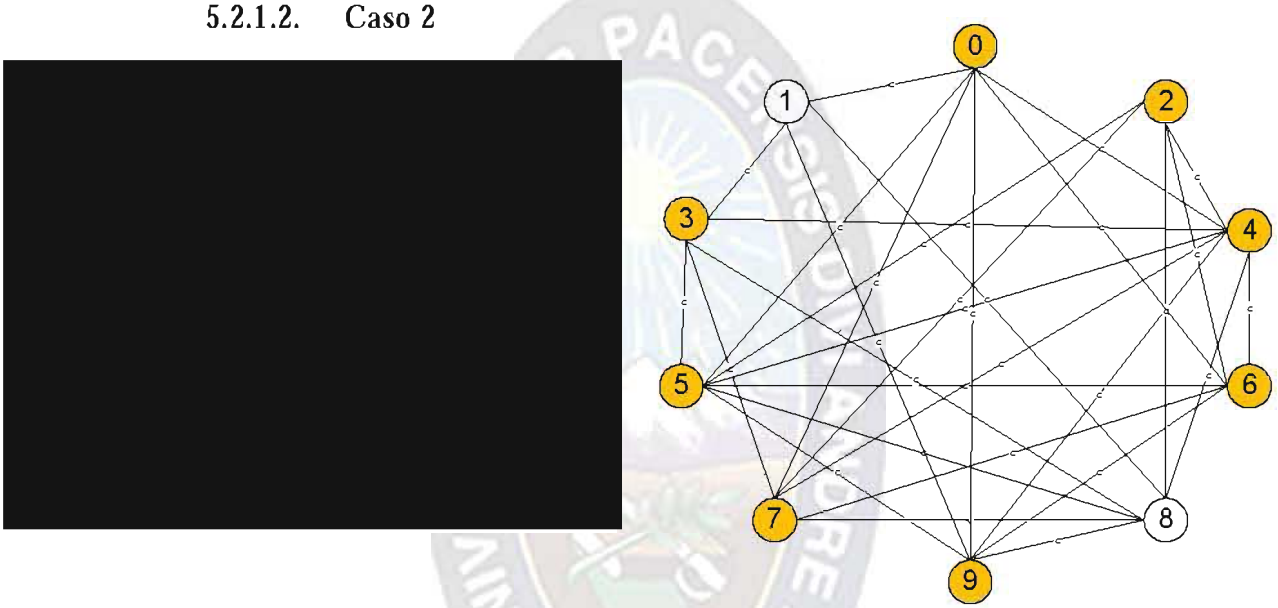


Figura 5.7 Grafo original para el caso de prueba 2

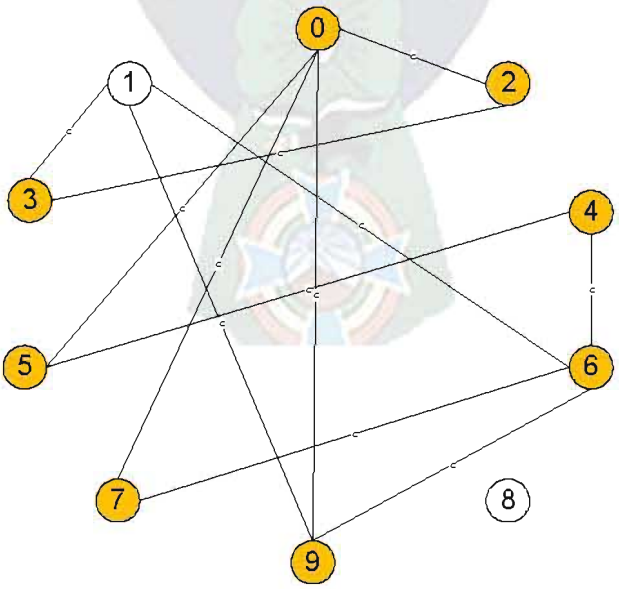


Figura 5.8 Grafo optimizado para el caso de prueba 2

[redacted]

Tabla 5.3 Resultado de la prueba sobre el caso 2

Los resultados que se observan en la Tabla 5.3 muestran aun con mayor claridad la optimización hecha a partir del grafo original respecto al grafo resultante. Hubo una reducción en el número de enlaces de 42 a 11, mientras que el costo total de la red se redujo de $ 79.64 a $ 16.63, el tiempo que le tomo al algoritmo encontrar estos resultados fue de 0.91 segundos.

5.2.1.3. Caso 3

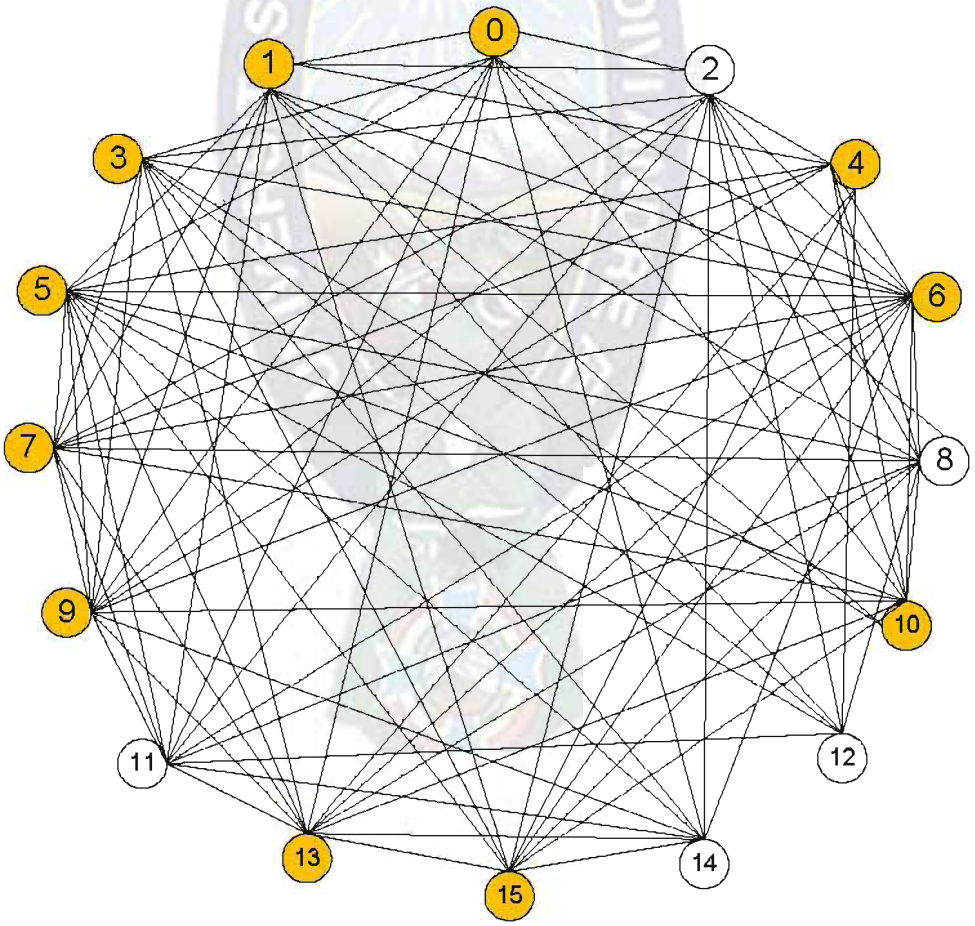


Figura 5.9 Grafo original para el caso de prueba 3

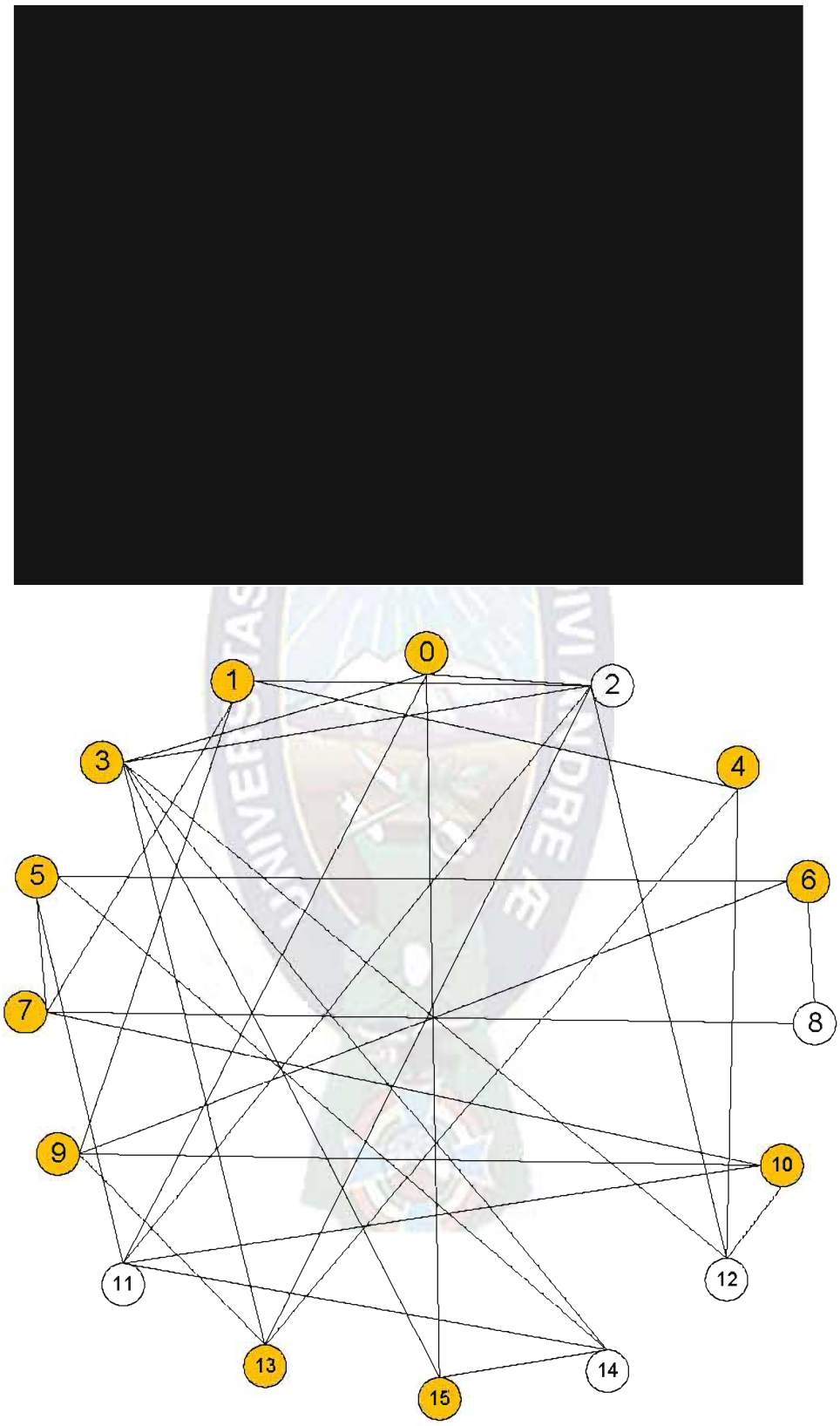


Figura 5.10 Grafo optimizado para el caso de prueba 2

Tabla 5.4 Resultado de la prueba sobre el caso 3

Para el último caso, los resultados que se observan en la Tabla 5.4 también muestran la optimización hecha a partir del grafo original respecto al grafo resultante. Hubo una reducción en el número de enlaces de 90 a 33, mientras que el costo total de la red se redujo de $ 215.09.64 a $ 65.87, el tiempo que le tomo al algoritmo encontrar estos resultados fue de 2.83 segundos.

5.2.2. Revisión de los tiempos de proceso

Las pruebas mostraron valores óptimos y cerca del óptimo global como se puede apreciar en la tabla 4.1, pero también tiempos cortos de procesamiento. Esto es algo que se espera de los algoritmos genéticos, pero que muestra que el modelo propuesto acelera la convergencia explorando la mayor parte del espacio de búsqueda.

La siguiente tabla muestra los tiempos de ejecución obtenidos con el experimento.

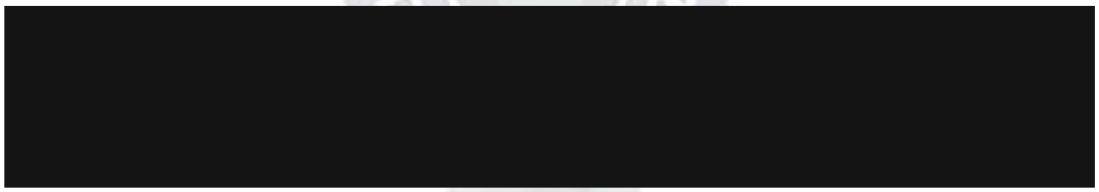


Tabla 5.5 tiempos de respuesta en pruebas

Se puede ver que los tiempos de respuesta tienen un comportamiento exponencial a medida que crece el espacio de búsqueda.

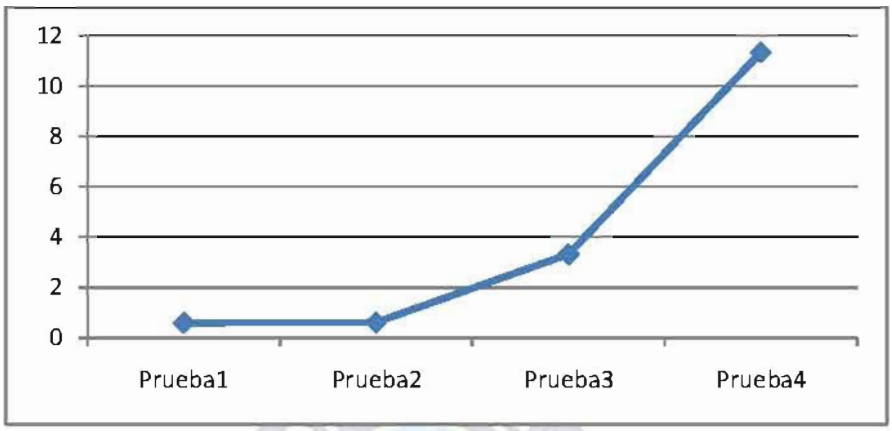


Figura 5.11 tendencia de los tiempos de respuesta

Finalmente no importa la cantidad de requerimientos o el número de veces que se ejecute el proceso, los tiempos de respuesta serán similares cuando se tiene un mismo espacio de búsqueda.

Para obtener los resultados de los caso de prueba se utilizo un equipo moderno ya que las comparaciones se realizaron en función del costo de la solución optima encontrada. El equipo tiene las siguientes características:

- Procesador Core 2 Duo de 2.5 GHz
- Memoria RAM de 2 GB
- Sistema Operativo Windows XP profesional SP3

CAPITULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

6. CONCLUSIONES Y TRABAJOS FUTUROS

RESUMEN

En este último capítulo se establecen las conclusiones a las que se ha llegado en relación a la totalidad de la investigación, se hace un análisis de los objetivos planteados y su nivel de satisfacción también se establece la evaluación de la hipótesis, se sugieren algunas recomendaciones relacionadas al estudio y se concluye con el planteamiento de trabajos futuros.

6.1. CONCLUSIONES GENERALES

Los algoritmos genéticos han demostrado su capacidad para resolver problemas del tipo planteado ya que estos se basan en la idea de que la evolución es un proceso de optimización que puede ser simulado en un computador y que mediante la emulación de los procesos naturales, principalmente el cruzamiento, la mutación, la competencia y la reproducción pueden resolverse problemas de optimización.

Los resultados obtenidos muestran alentadoras perspectivas sobre la aplicación de estas técnicas evolutivas para la resolución de problemas de optimización relacionadas con el diseño de redes de comunicación confiables.

El modelo planteado se mostro robusto en las pruebas de validación obteniendo muy buenos resultados en los tiempos esperados.

El algoritmo genético demostró que escogiendo una adecuada configuración de los parámetros de cruce y mutación es posible encontrar soluciones de calidad comparables a los algoritmos exactos. Si bien no siempre concluyen en el optimo global, lo compensan entregando soluciones optimas en tiempos más cortos.

Respecto a la performance los análisis demostraron que se puede obtener mejoras significativas de eficiencia utilizando un número pequeño de poblaciones y pocos individuos seleccionados.

La inclusión de conocimiento específico del problema, así como la adaptación de las estrategias heurísticas, mejoran el desempeño de manera muy importante. Tanto así que podemos decir que dada la necesidad de encontrar una solución a un problema real de tipo combinatorio, es muy recomendable la adaptación y combinación de los algoritmos genéticos con los diferentes métodos ya conocidos.

6.2. CUMPLIMIENTO DE OBJETIVOS

De acuerdo con los objetivos que se había planteado en el capítulo I, podemos concluir lo siguiente:

En el capítulo II, se ha hecho un estudio de las diferentes técnicas para resolver el problema planteado, tanto determinísticas como heurísticas, pasando a los algoritmos genéticos descritos en el capítulo III y como se aplican en la solución del problema, cumpliendo el objetivo de:

- Estudiar las diferentes alternativas para solucionar problemas de tipo combinatorio relacionados con el objetivo principal planteado, haciendo énfasis en las técnicas que aporta las tecnologías evolutivas

En el capítulo IV, se describe en detalle y aplica un algoritmo genético que resuelve el problema planteado, como nos habíamos propuesto en:

- Aplicar una técnica evolutiva que logre resolver el problema de Steiner en un grafo no orientado que representa geográficamente la topología de una red

Con la aplicación del algoritmo genético a la resolución del problema generalizado de Steiner, tal cual hemos demostrado y probado en el capítulo IV, hemos logrado cumplir con el objetivo:

- Interconectar un conjunto de nodos, geográficamente distribuidos, mediante una topología de red de bajo costo con capacidad de proveer una confiabilidad superior a la mínima especificada en los parámetros de diseño

Por último se ha podido validar y comparar los resultados obtenidos con el algoritmo genético propuesto frente a los ya conocidos, como habíamos planteado en:

- Analizar los resultados del AG en comparación con los otros métodos

6.3. ESTADO DE LA HIPOTESIS

Por las conclusiones obtenidas en el cumplimiento de objetivos las cuales nos ayudan a evaluar la hipótesis planteada, se concluye que:

Efectivamente es posible resolver el problema generalizado de Steiner para el diseño de una red de comunicación confiable y de bajo costo mediante una técnica evolutiva como método eficiente.

6.3.1. Recomendaciones

a) En el presente trabajo se eligió trabajar con poblaciones pequeñas, es decir pocos individuos, lo que reduce el espacio de búsqueda. Es recomendable utilizar poblaciones más grandes para ampliar la selección de individuos con diferentes características.

b) Es muy importante estudiar los diferentes métodos para cada operador, ya que si bien no se puede decir que uno sea mejor que otro, en la aplicación a un problema específico puede hacer la diferencia en términos de optimización o de encontrar la solución más cercana al óptimo global.

c) En el trabajo realizado se combinó las estrategias evolutivas con métodos conocidos de búsqueda de caminos. Es recomendable siempre que se pueda, hacer este tipo de combinaciones con técnicas que ya han demostrado su funcionalidad en problemas ya conocidos.

d) Dentro de las técnicas evolutivas existen otros operadores que no se han utilizado en esta solución, se recomienda su estudio y análisis para una posible implementación.

e) Se ha utilizado un algoritmo genético simple, pero la naturaleza como la evolución se vale de algunas otras ventajas como el paralelismo, esto puede mejorar o no el rendimiento, se recomienda su aplicación

6.4. TRABAJOS FUTUROS

a) Estudiar otro tipo de representaciones genéticas, ya que si bien la binaria hace más sencilla la mutación y el cruzamiento, es necesario corregirlas para encontrar más soluciones factibles.

b) Por su natural adaptación al paralelismo, estudiar e implementar estrategias evolutivas en paralelo, ya que prometen mejores resultados.

c) Implementar los modelos genéticos mencionados sobre estructuras en paralelo, tanto en clústeres como en supercomputadoras.

REFERENCIAS BIBLIOGRÁFICAS

REFERENCIAS BIBLIOGRÁFICAS

LIBROS

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Holland, J. H. (1999). *Adaptation in Natural and Artificial Systems.* NetLibrary.

Lahoz-Beltrá, R. (2004). *Bioinformática: Simulación, vida artificial e inteligencia artificial.* Ediciones Díaz de Santos.

ARTÍCULOS

Benjamín Barán, F. L. (2000). Diseño Económico de Redes confiables empleando A-Teams. *Centro Nacional de Computación, Universidad Nacional de Asunción .*

Campos Y., R. A. (2002). Algoritmo Evolutivo para el Problema de Árbol de Mínima Expansión MST.

Jose Hidalgo, C. C. (2002). Una revision de los algortimos genticos y sus aplicaciones. *Universidad Complutense de madrid .*

Londoño, N. G. (2006). Algoritmos Genéticos. *Universidad Nacional de Colombia, Escuela de Estadística .*

Magnago, H. E. (2006). Algoritmos evolutivos en problemas de diseño de redes. *Universidad nacional de la Plata .*

Nesmachnow, S. (2002). Diseño de redes de comunicaciones confiables - El problema de Steiner generalizado. *Universidad de la república de Uruguay .*

Robledo, F. (2000). Diseño topológico de redes, caso de estudio: " Problema Generalizado de Steiner" and "The Steiner 2-Edge-Connected subgraph problem". *Universidad de la república de Uruguay .*

Nesmachnow, S. (2002). Técnicas evolutivas aplicadas al diseño de redes de comunicaciones confiables. *Universidad de la república de Uruguay .*

Nesmachnow, S. (2002). Resolucion del problema de Steiner generalizado utilizando un Algoritmo genético paralelo. *Universidad de la república de Uruguay .*

Nesmachnow, S. (2003). Estudio Empírico de Operadores de Cruzamiento en un Algoritmo Genético Aplicado al Problema de Steiner Generalizado. *Universidad de la república de Uruguay* .

SITIOS WEB

Comisión Sectorial de Investigación Científica de Uruguay. (s.f.). *Algoritmos Genéticos Paralelos y su Aplicación al Diseño de Redes de Comunicaciones Confiables.* Recuperado el 2006, 2007, 2008, de http://www.fing.edu.uy/~sergion/gp.html

Dolan, A. (s.f.). *Artificial life and other experiments.* Recuperado el 2006, 2007, 2008, de http://www.aridolan.com/

Enrique Alba, Universidad de Malaga. (s.f.). *Lenguajes y ciencias de la computacion.* Recuperado el 2006, 2007, 2008, de http://www.lcc.uma.es/~eat/publi.html

Instituto de Computación y del Area Informática del PEDECIBA. (s.f.). *Biblioteca PEDECIBA y Reportes Técnicos.* Recuperado el 2008, de http://www.fing.edu.uy/inco/pedeciba/bibliote/reportes.html

ANEXO A

CÓDIGO FUENTE DEL ALGORITMO GENÉTICO

MÓDULOS

```
Option Explicit
Public nodo As Integer  ' numero de nodos
Public N As Integer ' cadena combinaciones
Public W As Single  ' pesos
Public W0 As Single ' pesos
Public W1 As Single ' pesos
Public pad1(500) As Integer ' padre1
Public pad2(500) As Integer ' padre2
Public m(100, 100) As Single ' matriz de pesos de los caminos
Public pop(4, 1000) As Integer  ' matriz de represetacion
Public a As Long  ' para el calculo del alea
Public MBIG As Long ' para el calculo del alea
Public MSEED As Long ' para el calculo del alea
Public MZ As Integer ' para el calculo del alea
Public FAC As Single ' para el calculo del alea
Public Ftbld As Integer  ' factibilidad
Public graph(100, 100) As Integer  'matriz de capacidades para maxflow
Public queue(100) As Integer  'cola en maxflow
Public head, tail As Integer  'variables de maxflow
Public parent(100) As Integer  'vector maxflow
Public V  'vertices en maxflow
Public s As Integer  'origen
Public t As Integer  'destino
Public fTotal As Integer  'flujo total
Public F(100, 100) As Long  'matris de flujos en maxflow
Public Cini As Single  'costo inical

Public Sub maxflow()
  Dim vj As Integer
  Dim min As Integer
  fTotal = 0
  While (reachable()) '(1 = reachable(s, t))
    'Gets the minimum possible capacity in edges of the path s to t
    min = graph(parent(t), t)
    vj = t
    While (parent(vj) <> vj)
      If (graph(parent(vj), vj) < min) Then
      min = graph(parent(vj), vj)
      End If
      vj = parent(vj)
    Wend
    vj = t
    While (parent(vj) <> vj)
      graph(parent(vj), vj) = graph(parent(vj), vj) - min
      graph(vj, parent(vj)) = graph(vj, parent(vj)) + min
      F(parent(vj), vj) = F(parent(vj), vj) + min
      vj = parent(vj)
    Wend
    fTotal = fTotal + min
  Wend
End Sub
'Breadth First Search
```

```
Public Function reachable() As Integer  '(s1 As Integer, t1 As Integer) As Integer
  Dim found As Integer
  Dim vq As Integer
  Dim i As Integer
  found = 0 'false
  head = 0
  tail = 0
  'memset(parent, 255, sizeof(parent))
  For i = 0 To V '(nodo-1)
  parent(i) = -1
  Next
  queue(tail) = s
  tail = tail + 1
  parent(s) = s
  While (head < tail And 1 = (1 + found))
    vq = queue(head)
head = head + 1
    For i = 0 To V '(nodo - 1)
    'Parents also made the function as visit vector
      If (graph(vq, i) And parent(i) = -1) Then
        queue(tail) = i
        tail = tail + 1
        parent(i) = vq
        If (i = t) Then
          found = 1 'true
          Exit For
        End If
      End If
    Next
  Wend
  reachable = found
End Function

Public Function alea(ByRef idnum As Long) As Single
Static inext As Integer
Static inextp As Integer
Static ma(56) As Long     '/* The value 56 (range ma[1..55]) is special */
                          '/* and should not be modified; see Knuth.    */
Static iff As Integer
Dim mj, mk As Long
Dim i, ii, k As Integer
If (idnum < 0 Or iff = 0) Then '/* Initialization */
  iff = 1     '/* Initialize ma[55] using the seed idnum and the large number MSEED */
  If idnum < 0 Then
    mj = MSEED + idnum
  Else
    mj = MSEED - idnum
  End If
  mj = mj Mod MBIG
  ma(55) = mj
  mk = 1
  '/* Now initizalize the rest of the table, in a slightly      */
  '/* random order, with numbers that are not especially random. */
  For i = 1 To 54
```

```
      ii = (21 * i) Mod 55
      ma(ii) = mk
      mk = mj - mk
      If mk < MZ Then
        mk = mk + MBIG
      End If
      mj = ma(ii)
    Next
    '/* We randomize them by "warming up the generator." */
    For k = 1 To 4
      For i = 1 To 55
        ma(i) = ma(i) - ma(1 + (i + 30) Mod 55)
        If ma(i) < MZ Then
          ma(i) = ma(i) + MBIG
        End If
      Next
    Next
    inext = 0      '/* Prepare indices for our first generated number. */
    inextp = 31    '/* The constant 31 is special; see Knuth */
    idnum = 1
End If
  '/* Here is where we start, except on initialization */
  inext = inext + 1   '/* Initizalize inext and inextp, wrapping    */
  If inext = 56 Then
    inext = 1
  End If
  inextp = inextp + 1 '/* arround 56 to 1.                          */
    If inextp = 56 Then
    inextp = 1
  End If
  mj = ma(inext) - ma(inextp)   '/* Generate new random number substractively */
  If mj < MZ Then
    mj = mj + MBIG  '/* Make sure that it is in range.            */
  End If
  ma(inext) = mj    '/* Store it                                  */
  alea = mj * FAC   '/* and output the derived uniform deviate.   */
End Function


Public Sub muta()
 Dim i As Integer
 Dim cam As Integer
 Dim x As Double
 'a = (94566 / Timer) * 58767
 x = Rnd
 If x <= 0.5 Then
    For i = 0 To ((N / 2) / 2) - 1
        cam = pad1(i)
        pad1(i) = pad2(i)
        pad2(i) = cam
    Next
 Else
```

```
    For i = nodo To ((N / 2) - 1)
        cam = pad1(i)
        pad1(i) = pad2(i)
        pad2(i) = cam
    Next
  End If
   For i = 0 To (N - 1)
    '13
    If i <= ((N / 2) - 1) Then
        pop(0, i) = pad1(i)
    Else
        pop(0, i) = pad2((i - (N / 2)))
    End If
  Next
  'mutacion adicional
  For i = 0 To (N - 1)
  x = Rnd
    If x < 0.1 Then
        If pop(0, i) = 1 Then
            pop(0, i) = 0
        Else
            pop(0, i) = 1
        End If
    End If
  Next
  End Sub

  Public Function fitness() As Single
    Dim i As Integer
    Dim sum As Single
    sum = 0
    For i = 0 To (N - 1)
        If pop(0, i) = 1 Then
            sum = sum + m(pop(1, i), pop(2, i))
        End If
    Next
    fitness = sum
  End Function

  Private Sub ruta()
  Dim i, j As Integer
  pop(3, 0) = 1
   For i = 0 To (N - 1)
    If pop(0, i) = 1 Then
        pop(3, i) = 1
        For j = (i - 1) To 0 Step -1
          If pop(2, j) = pop(2, i) Then
            If ((pop(0, j) = 1) And (pop(3, j) = 1)) Then
                If ((m(pop(1, j), pop(2, j))) >= (m(pop(1, i), pop(2, i)))) Then
                    pop(3, j) = 0
                    pop(3, i) = 1
                    j = 0
                End If
            End If
```

```
        Else
          pop(3, j) = 1
          pop(3, i) = 0
          j = 0
        End If
      Next
  End If
 Next
End Sub

' g = grado, R = requerimiento (1 cumple, 0 no cumple)
Public Function grado() As Integer
Dim i As Integer
Dim j As Integer
Dim g As Integer
Dim R As Integer
R = 1
g = 0
For i = 0 To (N - 1)
   If pop(3, i) <> 0 And R <> 0 Then
      g = 0
      For j = 0 To (N - 1)
         If pop(2, j) = pop(2, i) Or pop(1, j) = pop(2, i) Then
            If pop(0, j) = 1 Then
               g = g + 1
            End If
         End If
      Next
      If g >= pop(3, i) Then
         R = 1
      Else
         R = 0
         Exit For 'grado = R
      End If
   End If
Next
grado = R
End Function

' Fact = (1 cumple, 0 no cumple)
Public Function Factibilidad() As Integer
Dim i As Integer
Dim cg As Integer
Dim Fact As Integer

' Primero llenamos la matriz de capacidades de maxflow
For i = 0 To (N - 1)
   graph(pop(1, i), pop(2, i)) = pop(0, i)
   graph(pop(2, i), pop(1, i)) = pop(0, i)
Next
Fact = 1
'verifico grado de nodos terminales
cg = grado
If cg = 1 And Fact <> 0 Then
```

```
   For i = 0 To (N - 1)
      If pop(3, i) <> 0 Then
         s = pop(1, i)
         t = pop(2, i)
         Call maxflow
         If fTotal >= pop(3, i) Then
            Fact = 1
         Else
            Fact = 0
            Exit For 'Factibilidad = Fact
         End If
      End If
   Next
Else
Fact = 0
End If
Factibilidad = Fact
End Function


Public Sub main()
Dim i As Integer
Dim p As Integer
Dim c As Integer
Dim d As Integer
p = 0 ' peso
c = 0 ' corridas
d = 0 ' fallos
timer1 = Timer
'calaculo del costo de la generacion original
Cini = fitness
'For i = 0 To N - 1
'   If pop(0, i) = 1 Then
'      Cini = Cini + m(pop(1, i), pop(2, i))
'   End If
'Next
'ahora calculo la factibilidad de la inicial
Ftbld = Factibilidad
'cuento las aristas
'For i = 0 To (N - 1)
'   If pop(0, i) = 1 Then
'      p = p + 1
'   End If
'Next

'si es factible y "convergente", calculo fitness, grafico y termino
If Ftbld = 1 Then  'p = (nodo + 1) And Ftbld = 1 Then
   W0 = fitness
   W = Cini - W0
   'copio a optima
   For i = 0 To (N - 1)
     pop(4, i) = pop(0, i)
   Next
   ' aqui se grafica la red y el fitness
```

```
Else
   W = 0.01
   'copio a optima
   For i = 0 To (N - 1)
   pop(4, i) = pop(0, i)
   Next
End If
'empieza la evolucion
Do
d = d + 1
   '"seleccion" para cruce
   For i = 0 To (N - 1)
      If (i < (N / 2)) Then      ' tomar en cuenta que cambie el if <=
         pad1(i) = pop(4, i)
      End If
      If (i >= (N / 2)) Then     ' tomar en cuenta que cambie el if >
         pad2((i) - (N / 2)) = pop(4, i)
      End If
   Next
   'cruza y muta
   Call muta
   'calculo de factibilidad
   Ftbld = Factibilidad
      If Ftbld = 1 Then   'p = (nodo + 1) And Ftbld = 1 Then
      W0 = fitness
      W1 = Cini - W0
      If (W1 > W) Then
         W = W1
         c = 0
         'copio la solucion a optima
         For i = 0 To (N - 1)
         pop(4, i) = pop(0, i)
         Next
         ' aqui se grafica la red y el fitness
      Else
         c = c + 1
      End If
   End If
Loop While c <> 1000 And d <> 10000 '  lo mismo que while(c!=10);

timer2 = Timer - timer1
' aqui se grafica la red y el fitness
End Sub
```

FORMULARIOS

```
Private Sub CEvolucionar_Click()
Dim i As Integer
Dim j As Integer
Dim row As String
Dim row1 As String

'SETEO DE VARIABLES
V = nodo    ' Inicializacion para FF max flow
W = 0
W1 = 0

'Adicionales Alea
a = 203906
MBIG = 1000000000
MSEED = 161803398
MZ = 0
FAC = (1 / MBIG)

'Requerimientos de caminos
'pop(3, 6) = 2   ' requerimiento
'pop(3, 7) = 2   ' requerimiento
'pop(3, 14) = 2   ' requerimiento
'pop(3, 20) = 2   ' requerimiento
'pop(3, 25) = 2   ' requerimiento

'Llena la matriz de distancias
For i = 0 To nodo
   For j = 0 To nodo
      m(i, j) = Rnd * 5
      m(j, i) = m(i, j)
   Next
Next

Call main

'Impresion
For i = 0 To N - 1
   row1 = row1 & pop(3, i) & ";"  'concatena
   row = row & pop(4, i) & ";"  'concatena
Next

Form1.llegada.Text = row1
Form1.optima.Text = row
TCostoOriginal = Cini
TCostoOptimo = Cini - W
Print timer2

End Sub

Private Sub CRegistrar_Click()
Dim i As Integer
Dim j As Integer
```

```
Dim aux As Integer
Dim ori As Integer
Dim des As Integer
Dim req As Integer
ori = TORI
des = TDES
req = TREQ

If ori > des Then
   aux = des
   des = ori
   ori = aux
End If

For i = 0 To (N - 1)
   If pop(1, i) = ori And pop(2, i) = des Then
      pop(3, i) = req
   End If
Next
TORI = ""
TDES = ""

End Sub

Private Sub CLlenar_Click()
Dim x As Single
Dim i As Integer
Dim j As Integer
Dim c As Integer
Dim d As Integer
Dim e As Integer
Dim k As Integer
Dim row As String
Dim row1 As String
Dim row2 As String
row = ""
row1 = ""
row2 = ""

nodo = Tnodo - 1
N = (nodo * (nodo + 1)) / 2
'If ((nodo * (nodo + 1)) / 2) Mod 2 = 0 Then
'   N = ((nodo * (nodo + 1)) / 2)
'   Else
'   N = ((nodo * (nodo + 1)) / 2) + 1
'End If

'Llenar la primera fila de pop (existencia de arco)
For i = 0 To (N - 1)
   x = Rnd
   If (x > 0.15 And x < 0.8) Then
      pop(0, i) = 1
   Else
      pop(0, i) = 0
```

```
    End If
    row = row & pop(0, i) & ";" 'concatena los valores de existe
Next
Form1.existe.Text = row

'Llena la segunda y tercera fila de pop (desde,hasta)
e = nodo - 1
k = 0
c = 0
row = ""
For i = 0 To (nodo - 1)
    d = i + 1
    For j = k To (k + e)
        pop(1, j) = i
        pop(2, j) = d

        row1 = row1 & pop(1, j) & ";" 'concatena
        row2 = row2 & pop(2, j) & ";" 'concatena

        c = c + 1
        d = d + 1
    Next
    k = c
    e = e - 1
Next
Form1.desde.Text = row1
Form1.hasta.Text = row2

End Sub

Private Sub Form_Load()
Show
Tnodo = ""
Randomize (Timer)
End Sub
```

ANEXO B

COMPLEMENTOS Y PRUEBAS DE VALIDACIÓN

PROTOTIPO EXPERIMENTAL GENÉTICO

El prototipo experimental genético, crea una red inicial con enlaces y costos aleatorios, en base al tamaño ingresado en la pantalla de inicio. Los parámetros de operación han sido configurados a priori según la experiencia en el estudio del problema.

Se pueden ingresar los requisitos de comectividad necesarios de forma manual. Posterior a ello se puede iniciar la evolución y una vez convergido el AG devuelve la red resultante junto con los costos optimizados.

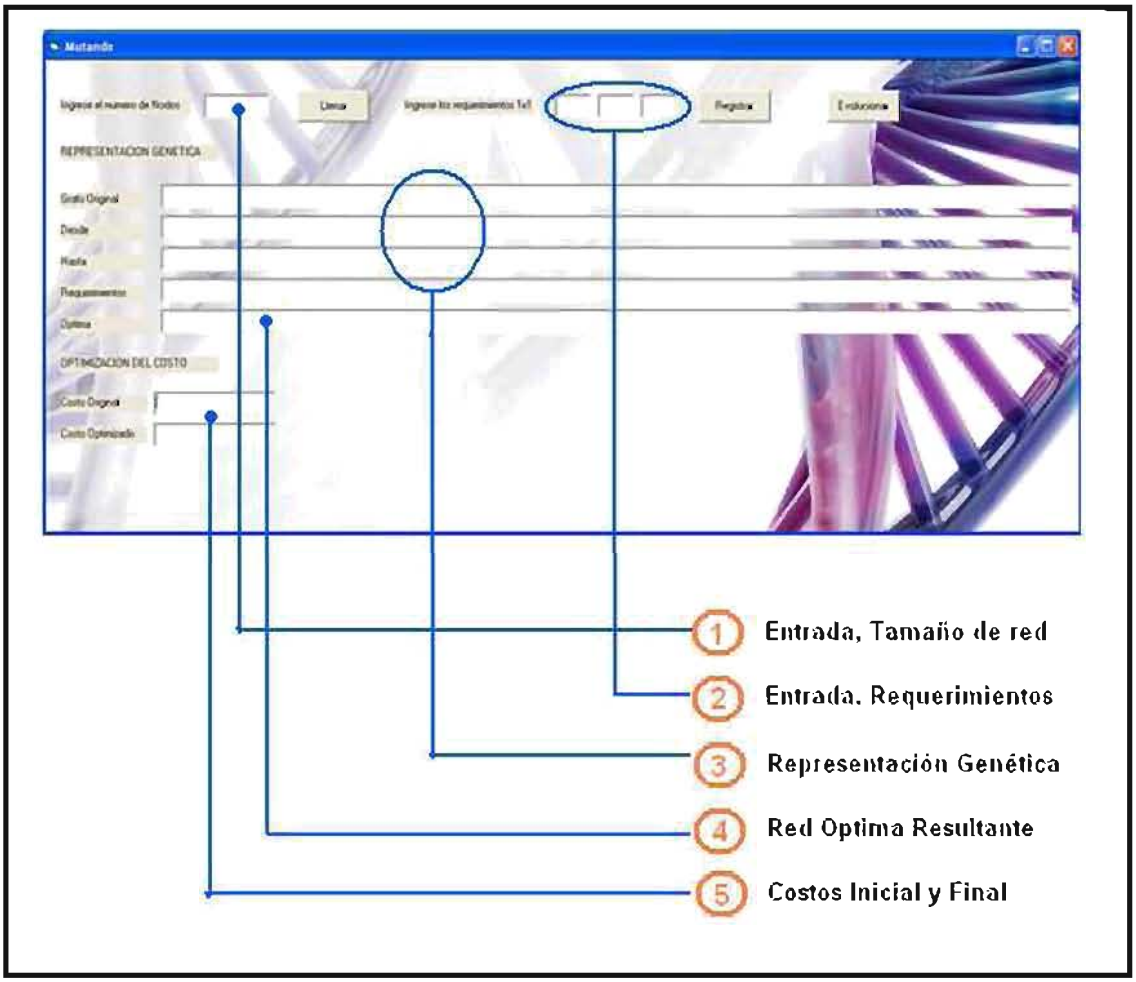


Figura 1: Pantalla de inicio, prototipo experimental genético

REPRESENTACIÓN Y RESULTADOS PRUEBA 1

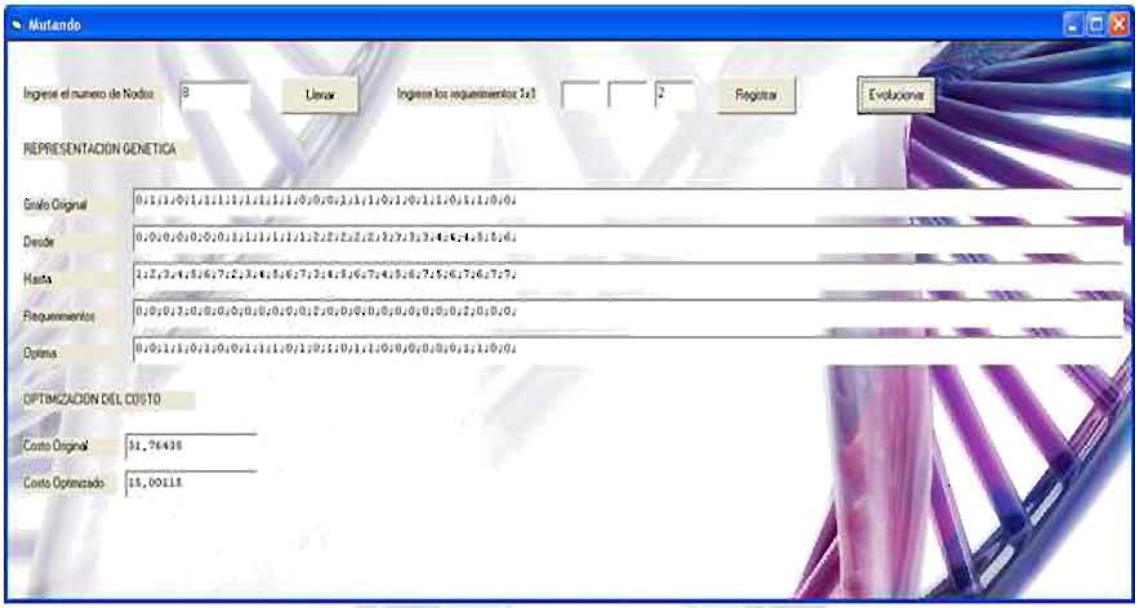


Figura 2: Representación y resultados prueba 1

REPRESENTACIÓN Y RESULTADOS PRUEBA 2

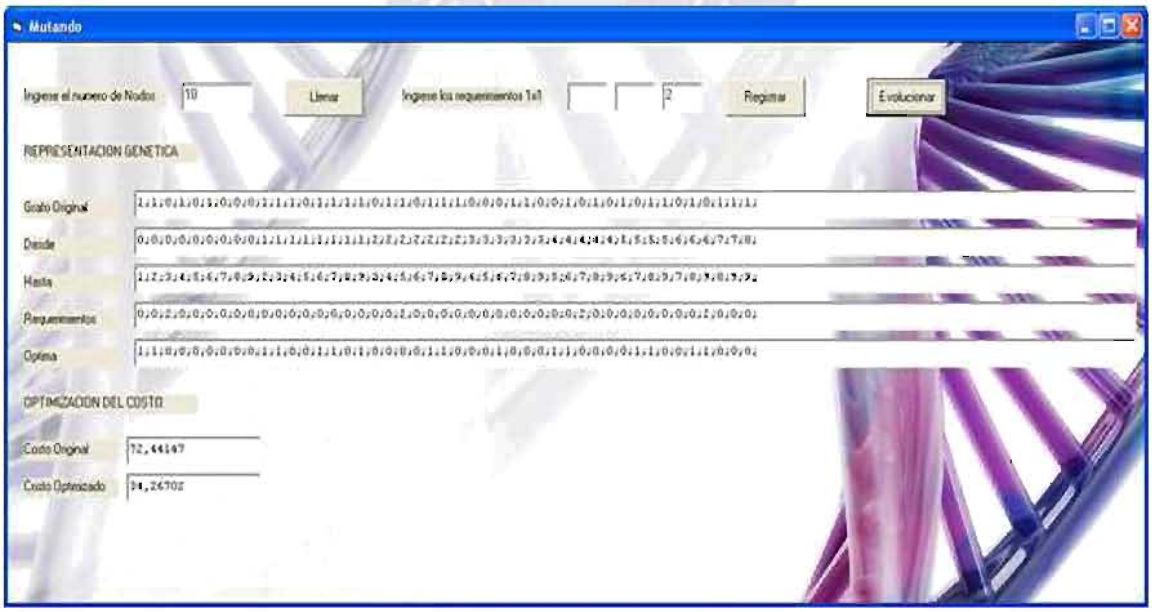


Figura 3: Representación y resultados prueba 2

REPRESENTACIÓN Y RESULTADOS PRUEBA 3

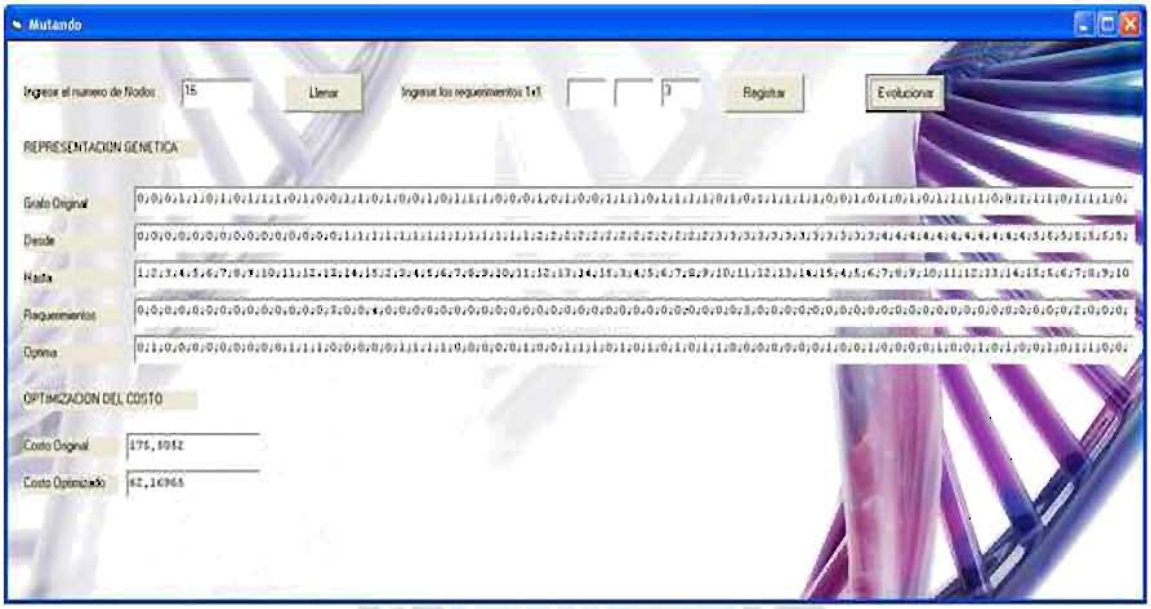


Figura 4: Representación y resultados prueba 3