Optimización de la confiabilidad global de redes generales mediante la asignación de arcos

por
Jesús Andrés Rodríguez Sarasty

Tesis sometida en cumplimiento parcial
de los requisitos para el grado de

MAESTRO EN CIENCIAS
en
Ingeniería Industrial

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ
2012

Aprobada por:

_____	_____
Mercedes Ferrrer Alameda, MSc Miembro del Comité Graduado	Fecha
_____	_____
Noel Artiles-León , PhD Presidente Comité Graduado	Fecha
_____	_____
Anand D. Sharma, PhD Representante de Estudios Graduados	Fecha
_____	_____
Viviana Cesaní, PhD Miembro del Comité Graduado	Fecha
_____	_____
Viviana Cesaní, PhD Director del Departamento	Fecha

Abstract

In general networks, the redundancy allocation problem (RAP) consists in allocating a number of links to each connection of a network, whether to minimize the cost of the system (subject to a reliability constraint), or to maximize the network reliability (subject to a total budget).

Considering the computational complexity for the network reliability calculation and for the solution of the optimization problem, in this work we propose and compare three heuristics, which use efficient techniques for network connectivity evaluation, network reduction, reliability evaluation and solution finding. In computational experiments, the heuristic with the reliability upper bound outperformed the other two heuristics in terms of running time.

Resumen

En redes generales, el problema de asignación de redundancias (*redundancy allocation problem, RAP*) consiste en determinar el número de arcos que se deben asignar a cada conexión de la red, bien sea para minimizar el costo del sistema, sujeto a una restricción de confiabilidad, o para maximizar la confiabilidad de la red, dado un presupuesto total.

Considerando la complejidad computacional para el cálculo de la confiabilidad de la red, y para la solución del problema de optimización, en este trabajo se proponen y comparan tres heurísticos que utilizan técnicas eficientes para la evaluación de la conectividad de redes, la reducción de la red, la evaluación de la confiabilidad y la búsqueda de soluciones. En experimentos computacionales, el heurístico con la cota superior de confiabilidad superó a los otros dos heurísticos en términos de tiempos de ejecución.

Agradecimientos

Agradezco a los profesores y al personal administrativo del Departamento de Ingeniería Industrial, por todo el apoyo y la colaboración que me han brindado.

También expreso mi agradecimiento a los profesores miembros del Comité Graduado, Viviana Cesaní, Anand Sharma y Mercedes Ferrer por sus comentarios y sugerencias para mejorar este documento.

Mi sincero agradecimiento al profesor Noel Artiles por su paciencia, sus orientaciones y su colaboración, no solo para la realización de esta investigación, sino también para que yo pudiera cursar y completar mis estudios en la Universidad de Puerto Rico.

Tabla de contenido

Lista de Tablas

Lista de Figuras

1 Introducción

1.1 Motivación

A través de las redes ocurre el flujo de materiales, energía e información que la sociedad demanda. En situaciones reales, el diseño de redes exige considerar multitud de criterios, tales como la inversión, el costo de operación, la capacidad, la velocidad y la confiabilidad del sistema. En particular esta última es una característica de diseño de la que no sólo depende la calidad del producto y el funcionamiento del sistema, sino incluso la vida de las personas. Por ejemplo, una falla inesperada en una red de transmisión de gas podría poner en riesgo la vida de los habitantes del sector aledaño al lugar del accidente.

Aunque la creciente dependencia tecnológica incrementa la exigencia de confiabilidad de los sistemas, en la práctica existen dificultades para el análisis y diseño de redes confiables. Esto no sólo se debe a la cada vez mayor complejidad de los sistemas, sino también a la limitada aplicabilidad de algunos de los métodos para el análisis de confiabilidad de redes, debido a su inherente complejidad computacional. Por tal razón, se hace necesario avanzar en el desarrollo de métodos que permitan llevar a la práctica el análisis y diseño de sistemas confiables.

En este orden de ideas, en este trabajo analizan y proponen métodos para maximizar la confiabilidad global de redes generales, considerando una restricción presupuestal.

1.2 Conceptos iniciales

La **confiabilidad** es la probabilidad de que un sistema se desempeñe adecuadamente durante un determinado periodo, cuando es operado en las condiciones especificadas (Kuo & Zuo, 2003, p. 1).

En una red, la **confiabilidad global** (generalmente referida en inglés como ¨*all-terminal reliability*¨, ¨*overall reliability*¨ o ¨*global reliability*¨), es la probabilidad de que la red esté conectada, es decir, la probabilidad de que exista al menos una ruta entre cada par de nodos (Cormen, Leiserson, & Rivest, 1990)**.** En diversos sistemas reales, y particularmente en redes de comunicación, el funcionamiento del sistema depende de su estado de conectividad (Wilkov, 1972, p. 660), es decir, de la posibilidad de establecer conexión con los demás nodos de la red.

En **redes generales**, es decir, en redes que pueden tener cualquier configuración, la evaluación de la confiabilidad global es un problema *NP-hard* (Ball, 1986, p. 236). Para ilustrar la complejidad del cálculo de la confiabilidad, considérese una **red completamente conectada**, es decir, una red en la cual existe una conexión entre cada par de nodos de la red. En esta red, el número m de conexiones es,

$m = \mathbb{C}_2^n = n(n-1)/2$, donde n es el número de nodos. Si cada arco tiene dos estados posibles, el total de posibles estados del sistema es $2^{n(n-1)/2}$. Por lo tanto, en una red con 20 nodos completamente conectados hay 1.56927×10^{57}estados. Al asumir que la evaluación de cada estado de la red requiere una cantidad de operaciones igual al número de conexiones, en una red de este tamaño se requerirían 190 operaciones para cada uno de los posibles estados. De esta manera, un computador que ejecutara 100,000 millones de operaciones por segundo, requeriría 2.9816×10^{48} segundos para evaluar todos los estados. Esta magnitud de tiempo es aproximadamente 6.7487×10^{30} veces la edad del universo, la cual se estima en 14 mil millones de años (Knox, Christensen, & Skordis, 2001).

El diseño óptimo de una típica red real entraña complicaciones adicionales, dado que además de la confiabilidad se deben considerar otros factores tales como costos, condiciones de operación y requerimientos funcionales (Shooman, 2002, p. 310). Por tal razón, las investigaciones en el área han buscado desarrollar métodos aproximados de solución o analizan versiones simplificadas del problema, considerando un limitado subconjunto de criterios de diseño (Colbourn, 1999 , p. 141).

1.3 El problema

De las múltiples medidas existentes para incrementar la confiabilidad de redes generales, en este estudio solamente se considera la opción de asignar componentes (arcos), algunos de ellos redundantes, cuyos costos y confiabilidades dependen de la conexión donde se asignen.

Este problema cae en la categoría del problema de asignación de redundancias, RAP, por su nombre en inglés (*Redundancy Allocation Problem*), el cual aún para el simple caso de asignación de redundancias en un sistema en serie con restricciones lineales es *NP-completo* (Sung *et al.*,(2003, p. 93). Para el RAP con dos terminales se han propuesto distintos métodos de solución basados en heurísticos, metaheurísticos y técnicas de optimización clásica [Kulturel-Konak*et al.*, (2003); Ha & Kuo (2006); Tavakkoli-Moghaddam*et al.* (2008); Billionnet(2008); Kim & Yum (1993); Liang & Chen (2007); Ramirez-Marquez *et al*. (2004); Yalaoui*et al*. (2005), Hernández (2007)].

A diferencia de los trabajos previos, en esta investigación se asume que debe existir conectividad entre todos los nodos de la red. Por lo tanto, se busca proponer algoritmos para determinar el número de arcos que deben asignarse a cada conexión de una red general, de manera que se maximice la confiabilidad global, a la vez que se cumple con una restricción presupuestaria y sin exceder las cantidades máximas y mínimas de arcos que pueden asignarse a cada posible conexión. En otras palabras, el problema de optimización a resolver es:

$$\text{Maximizar} \quad R_s(x_1, x_2, \dots, x_i, \dots, x_m)$$

Sujeto a

$$\sum_{i=1}^{m} c_i x_i \leq b$$

$$x\_min_i \leq x_i \leq x\_max_i \forall i$$

$$x_i \in \mathbb{Z}\ \forall i$$

$$x_i \geq 0 \quad \forall i$$

Donde,

m : Total de posibles conexiones de la red

x_i : Número de arcos que deben asignarse a la conexión i

c_i : Costo de cada nuevo arco asignado a la conexión i

b : Presupuesto total disponible para los nuevos arcos

x_min_i : Número mínimo de nuevos arcos en la i-ésima conexión, $x\_min_i \geq 0$

x_max_i : Número máximo de nuevos arcos en la i-ésima conexión

$R_s(x_1, x_2, \dots, x_i, \dots, x_m)$: Función de confiabilidad del sistema.

En teoría esta función puede calcularse de manera exacta, mediante productos y sumas de términos de la forma,

$$p_i = 1 - (1 - r_{0i})^{q_i}(1 - r_{Ai})^{x_i} \tag{1.1}$$

Donde,

p_i: Confiabilidad de la conexión i

r_{0i} : Confiabilidad de cada uno de los arcos de la red original (preexistentes) en la conexión i

r_{Ai} : Confiabilidad de cada nuevo arco asignado a la conexión i

q_i : Cantidad de arcos de la red original (preexistentes) en la conexión i

Desde el punto de vista matemático se enfrentan varios inconvenientes para resolver este problema combinatorio. Además de la dificultad para el cálculo de la confiabilidad, "la función de confiabilidad es no lineal, no separable y no convexa" (Ha & Kuo, 2006, p. 25), razón por la cual no es viable utilizar directamente "técnicas eficientes de programación entera para la solución de problemas enteros lineales, separables o convexos" (Ha & Kuo, 2006, p. 25).

1.4 Supuestos

Para simplificar el problema sin pérdida de generalidad, se establecen los siguientes supuestos:

1) En cada conexión factible se consideran dos tipos de arcos: los previamente instalados en la red original (**preexistentes**) y los que se van a agregar (**nuevos**). Cuando en ninguna de las conexiones hay arcos preexistentes se tiene el caso particular del diseño de una nueva red.
2) Los arcos únicamente tienen dos estados posibles: funcionan o fallan.
3) Los costos son conocidos.
4) Los nodos son perfectamente confiables.
5) Las fallas en los arcos ocurren de manera independiente.
6) Los arcos son bidireccionales.
7) No se hacen reparaciones.
8) Las redundancias son activas.
9) Las confiabilidades de los arcos son conocidas.
10) El sistema es coherente.

Este último supuesto implica las siguientes condiciones:

i.) La confiabilidad del sistema no disminuye ante el mejoramiento de la confiabilidad de los arcos o la adición de arcos en paralelo. Aunque esta condición típicamente se cumple en los sistemas reales (Colbourn, 1999 , p. 138), en algunos sistemas la confiabilidad global del sistema puede disminuir al agregar componentes redundantes más allá de una cierta cantidad. Por ejemplo, en un cohete de lanzamiento, la inclusión excesiva de componentes redundantes puede incrementar significativamente el peso, lo cual demandaría mayor potencia durante el lanzamiento, y aumentaría el riesgo de falla.

ii.) Todos los componentes del sistema son relevantes. Es decir, para cada uno de los componentes se cumple que, en al menos una de las combinaciones de los componentes del sistema, el funcionamiento del componente determina la conectividad del sistema(Kuo & Zuo, 2003).

1.5 Objetivos

En concordancia con el problema definido, en este trabajo se va a:

i) Desarrollar un algoritmo eficiente para la determinación de la conectividad de redes generales.
ii) Implementar un método eficiente para evaluar la confiabilidad global de redes generales.
iii) Desarrollar tres métodos heurísticos para maximizar la confiabilidad global de redes generales mediante la asignación de arcos.
iv) Comparar el desempeño de los heurísticos desarrollados.

1.6 Trabajos previos

Algunas investigaciones en este campo se han enfocado a desarrollar métodos de solución exacta, tales como enumeración implícita, ramificación y acotamiento, y programación dinámica [Sung, Cho, & Song (2003), Kuo *et al.* (2001)]. Sin embargo, hasta el más eficiente de estos métodos es inadecuado para resolver problemas de mediano o gran tamaño (Ha & Kuo, 2006, p. 26). Debido a esta limitación, se han explorado tres enfoques de solución aproximada para este problema:

- Relajación entera: Se aplican algoritmos de programación matemática sin restringir el dominio de las variables a valores enteros. Luego, el resultado obtenido se aproxima a una solución entera (Ha & Kuo, 2006, p. 25).
- Heurísticas: Con base en las propiedades del espacio de solución del problema se diseñan algoritmos de búsqueda ajustados a la situación específica, los cuales alcanzan soluciones razonables, en algunos casos cuasi-óptimas, con un esfuerzo computacional significativamente menor que el requerido por los métodos exactos. Este enfoque es particularmente útil cuando la solución óptima exacta es poco relevante, ya que pueden existir errores y aproximaciones en los valores de las probabilidades y los costos (Ha & Kuo, 2006, p. 66).
- Metaheurísticas: Este enfoque proporciona una estructura iterativa que mediante estrategias de aprendizaje guía la exploración eficiente del espacio de solución, a la vez que evita el estancamiento en óptimos locales. Algunas metaheurísticas comunes son los algoritmos genéticos, la búsqueda tabú, las colonias de hormigas y el recocido simulado.

Aunque la mayoría de estos enfoques ha tenido cierto grado de éxito en la solución de casos particulares, ninguno de ellos se puede considerar como el más indicado para resolver cualquier problema de optimización de confiabilidad de redes (Kuo *et al.,*2001, p. 44). Por esta razón, se requiere el desarrollo de algoritmos que permitan resolver casos más generales de forma eficiente.

Algunos de los heurísticos relacionados con el problema de investigación, son los siguientes:

- Kontoleon (1979): Asigna una cantidad determinada de arcos con diferentes confiabilidades, a redes de dos terminales (es decir, un nodo origen y un nodo destino) con topología fija. Para asignar los arcos el algoritmo evalúa la variación de la función de confiabilidad de la red, con respecto a cada posible arco.
- Shi (1987): Optimiza la redundancia en redes de dos terminales. Mediante una razón de la confiabilidad entre el costo de los recursos, el método selecciona la ruta mínima de mayor sensibilidad; la redundancia se asigna al componente más sensible en dicha ruta.

- Kiu & McAllister (1988): Asigna arcos a una red de topología fija, con el fin de maximizar la confiabilidad global de la red. En cada iteración, el heurístico busca asignar los arcos más confiables a las conexiones más débiles.
- Fard & Lee (2001): Asigna arcos a la red de manera que se maximice el número de árboles de expansión. Aunque este heurístico es bastante práctico, y a pesar de que el número de árboles de expansión tiene una estrecha relación con la confiabilidad global de la red, las soluciones obtenidas mediante este heurístico pueden desviarse de la solución óptima debido a que no considera explícitamente las confiabilidades de los arcos.
- Hernández (2007): Utiliza programación lineal entera secuencial (PLES) para la asignación de componentes redundantes en redes de dos terminales. Se comparó el desempeño de este heurístico contra un método basado en algoritmos genéticos. En experimentos con redes con un máximo de 20 nodos y 190 arcos, el heurístico fue más rápido que el método basado en algoritmos genéticos (Hernández, 2007).

También se han aplicado diversos metaheurísticos en problemas de confiabilidad de redes. En particular, existen numerosos trabajos sobre la aplicación de algoritmos genéticos [Gen & Cheng, 2000; Get *et al.*, 2008]. Algunos trabajos similares a esta investigación, son

- Deeter & Smith (1997). Mediante algoritmos genéticos minimizan los costos de una red general de topología fija sin componentes redundantes, sujeta a una restricción de confiabilidad global. En cada conexión de la red se puede escoger entre arcos con diferentes costos y confiabilidades. Gen & Cheng (2000). Describen con detalle la implementación de este algoritmo.
- Altiparmak *et al*. (1998). Desarrollaron un algoritmo genético para un problema similar de topología fija de red. En cada conexión se puede escoger entre diferentes tipos de arcos, con distintos costos y confiabilidades, a fin de maximizar la confiabilidad global, considerando una restricción presupuestaria.
- Dengiz *et al*.(1997). Este método es explicado por Gen & Cheng (2000). Dada una red con arcos de igual confiabilidad, y con un costo que depende de la conexión donde se asignen, se busca encontrar el diseño topológico que maximice la confiabilidad global, cumpliendo con una restricción presupuestaria. A las redes candidatas que cumplen unas ciertas condiciones de conectividad, se les calcula la cota superior de confiabilidad, de acuerdo con el método de Jan (1993). Sólo las redes cuya cota superior sea mayor que la restricción de confiabilidad mínima, y cuyo costo sea menor que el de la mejor solución actual, se envían a la rutina para estimación de la confiabilidad mediante simulación Monte Carlo.

El lector puede referirse a Gen *et al*. (2008) para una discusión adicional sobre la aplicación de algoritmos genéticos a problemas de confiabilidad de redes.

Debido a que no parece existir un método claramente superior para resolver los problemas de optimización de confiabilidad (Kuo & Wan, 2007, p. 150), en esta investigación se desarrollan, analizan y comparan dos métodos de solución: un método basado en algoritmos genéticos, y una adaptación del heurístico propuesto por el Dr. Artiles-León e implementado por Hernández (2007) para la optimización de confiabilidad de redes con un nodo de origen y un nodo destino.

1.7 Organización del documento

Teniendo en cuenta que la conectividad es una condición indispensable de las soluciones potenciales, y que la confiabilidad está determinada por el valor esperado de la conectividad de la red, en el capítulo 2 se estudian las condiciones de conectividad, se evalúan varios métodos para su determinación y se desarrolla un algoritmo para la evaluación de conectividad de redes generales. En el capítulo 3 se discuten los principales enfoques para la evaluación de confiabilidad global y se desarrolla un algoritmo para la evaluación de la confiabilidad global de redes generales. En el capítulo 4 se describen los dos heurísticos propuestos para resolver el problema definido en la Sección 1.3. En el capítulo 5 se exponen los resultados de los experimentos computacionales realizados. Finalmente, se presentan las conclusiones y las recomendaciones para trabajos futuros en esta área.

2 Conectividad de redes generales

2.1 Generalidades

Dada la complejidad computacional del cálculo exacto de la confiabilidad (Ball, 1986, p. 236), en este trabajo se optó por utilizar métodos Monte Carlo para estimar la confiabilidad global. Con este enfoque la estimación de la confiabilidad en tiempos razonables depende en gran medida de la disponibilidad de un algoritmo eficiente para la determinación de la conectividad, dado que la simulación Monte Carlo estima la confiabilidad a partir del promedio del estado de conectividad de una muestra aleatoria de posibles estados de la red.

En este capítulo se analizan las condiciones necesarias y suficientes para la conectividad de redes generales no dirigidas. Luego se discuten varios métodos para la determinación de conectividad, y se hace una comparación empírica de los mismos, considerando diferentes tamaños de red. A partir del método con mejor desempeño y utilizando las condiciones de conectividad necesarias y suficientes, se desarrolla un algoritmo para la determinación de conectividad de redes generales.

2.2 Condiciones de conectividad necesarias y suficientes en redes generales no dirigidas

Dengiz *et al.*(1997) propusieron un algoritmo de optimización de confiabilidad de redes, en el cual evalúan que todos los nodos de las redes candidatas deben tener grado 2, es decir, que cada nodo debe estar conectado por lo menos con otros dos nodos de la red. Debido a que esta condición no garantiza la conectividad de la red y elimina posibles soluciones factibles y óptimas, se requiere explorar otras condiciones de conectividad. En esta sección de discuten dos condiciones necesarias de conectividad, y se deriva una condición suficiente. Las condiciones necesarias son aquellas que cumple toda red conectada, pero que al cumplirse no garantizan la conectividad de la red. Por otro lado, la condición suficiente es aquella que al cumplirse indica que la red está conectada, pero su incumplimiento no implica ausencia de conectividad.

2.2.1 Condiciones necesarias

Sea $G(C,V)$ una red G no dirigida con un conjunto C de conexiones y con un conjunto V de nodos o vértices, tal que $|V| = n$, el número de nodos y $|C| = m$, el número de conexiones. El grado g de los nodos se define como la suma de las conexiones que tienen contacto con el nodo. Si esta la red está conectada, entonces:

Lema 1: El grado de cada uno de los vértices de la red es mayor o igual que uno. Es decir,

$$g_i \geq 1, \quad \forall\, i \in V \tag{2.1}$$

***Prueba*:** Asúmase que el grado de al menos uno de los nodos es igual a cero. Puesto que este nodo no tiene conexión con ninguno de los nodos de la red, la red no está conectada. Por lo tanto, para que exista conectividad es condición necesaria que el grado de todos los nodos de la red sea mayor o igual que 1. ■

Lema 2: La suma del grado de los vértices de la red es mayor o igual que $2(n-1)$. Esto es,

$$\sum_{\forall\, i \in V} g_i \geq 2(n-1) \tag{2.2}$$

Prueba: Para que los n nodos de una red estén conectados debe haber por lo menos $n-1$ arcos. Dado que cada arco conecta dos nodos, para que exista conectividad es condición necesaria que el mínimo valor de la suma del grado de todos los nodos de la red sea mayor o igual que $2(n-1)$. ■

El cumplimiento de estas condiciones necesarias no garantiza la que la red esté conectada. Sin embargo, el incumplimiento de alguna de ellas implica que no hay conectividad.

2.2.2 Condición suficiente

Lema 3: Una red no dirigida está conectada si,

$$\sum_{\forall\, i \in V} g_i \geq n^2 - 3n + 4 \tag{2.3}$$

Prueba: En una red no conectada, al menos uno de los n nodos no tiene conexión con los $n-1$ nodosconectados restantes. El máximo número de arcos entre estos $n-1$ nodos conectados es $\binom{n-1}{2}$. Si el nodo no conectado se conecta con al menos uno de los nodos restantes, la red estará conectada y tendrá por lo menos $\binom{n-1}{2}+1$conexiones. Dado que la suma del grado de los nodos es igual al doble del total de las conexiones,

$$\sum_{\forall\, i \in V} g_i \geq 2\left[\binom{n-1}{2}+1\right]$$

Lo cual equivale a,

$$\sum_{\forall\, i \in V} g_i \geq n^2 - 3n + 4 \ \blacksquare$$

La Ecuación 2.3 se puede reescribir como,

$$D \geq 1 + \frac{4-2n}{n(n-1)} \tag{2.4}$$

Donde D es la densidad de red, la cual corresponde al número de conexiones dividido entre el total de posibles conexiones, es decir,

$$D = \frac{2m}{n(n-1)} \tag{2.5}$$

Donde m es el número de conexiones de la red.

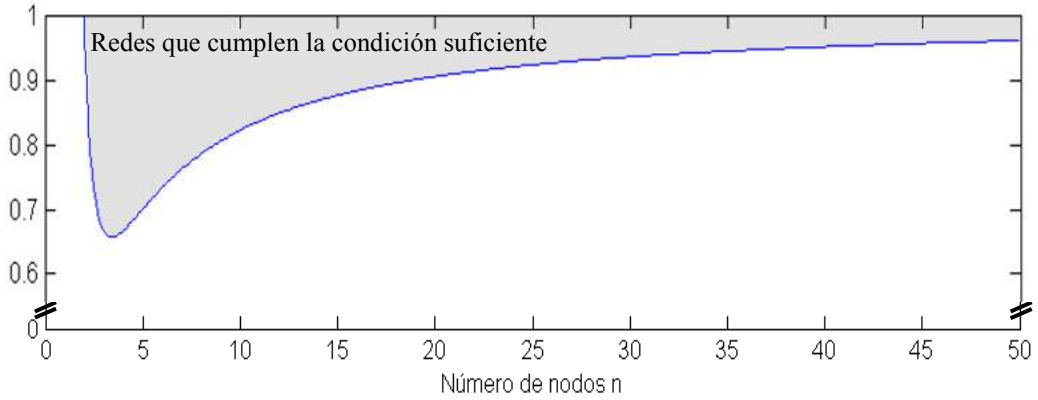

Figura 1.Densidades mínimas de red que cumplen la condición suficiente de conectividad, considerando diferentes tamaños de red

Es evidente la estrecha relación entre la densidad y la conectividad de la red. La densidad mínima para garantizar conectividad tiende asintóticamente a 1 a medida que incrementa el número de nodos de la red (Figura 1). De acuerdo con la Ecuación (2.4)**,** en redes con menos de 20 nodos y densidad mayor o igual que 0.9 se garantiza la conectividad.

2.3 Métodos para determinación de conectividad

Para la determinación de conectividad comúnmente se utilizan algoritmos de búsqueda en amplitud o búsqueda en profundidad (*breadth-first* o *depth-first*, respectivamente). Estos métodos requieren estructuras de árbol y realizan rastreo hacia atrás (*backtracking*) en un proceso iterativo que busca "descubrir" o conectar todos los nodos de la red. Por tal razón su implementación y desempeño en lenguajes de tipo matricial tales como Matlab es poco conveniente.

Se describen a continuación seis alternativas para la determinación de la conectividad: rutas mínimas entre todos los nodos, identificación de un árbol de expansión, conectividad algebraica, número de árboles de expansión, multiplicación de la matriz de conexión, y producto booleano de la matriz de adyacencia. Mediante la red de la Figura 2 se ejemplifica la aplicación de los métodos que a continuación se describen.

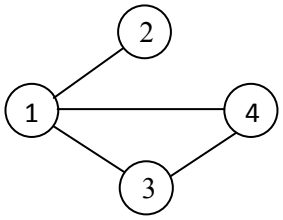

Figura 2. Red no dirigida

2.3.1 Rutas mínimas entre todos los nodos

La posibilidad de aplicar este método se basa en el hecho de que si existe una distancia finita entre cada par de nodos de la red, entonces la red está conectada. Existen diversos algoritmos para la determinación de las rutas mínimas entre todos los nodos de una red (Cormen *et al.,*1990). A continuación se explica el método propuesto por Robert Floyd en 1962. Para su implementación se siguió el desarrollo propuesto por Cormen *et al*. (1990, pp. 558-562).

Sean tres nodos *i*, *j*, *k*, como se muestra en la Figura 3.

.

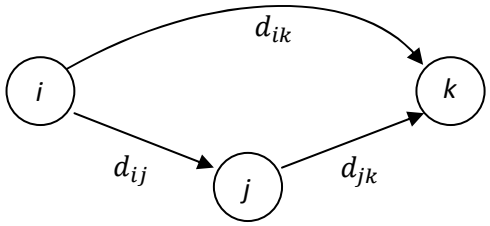

Figura 3.Patrón de comparación del algoritmo de Floyd

Se puede ir del nodo i al nodo k, mediante el arco d_{ik} que los conecta directamente, o siguiendo una ruta a través de otros nodos (en el ejemplo de la Figura 3, siguiendo la ruta d_{ij}, d_{jk}). Dependiendo de las distancias de cada uno de los arcos, la ruta más corta entre los nodos i, k puede ser aquella que visita varios nodos antes de llegar al nodo k.

En el algoritmo de Floyd, se inicia con la matriz de distancias entre los nodos. Si dos nodos no están conectados, se asigna un valor infinito a la distancia entre ellos. Se asume que ésta es la distancia mínima inicial. En cada iteración se actualiza el cálculo de la distancia mínima entre cada par de nodos, considerando la posibilidad de pasar por otro nodo de la red. La distancia mínima desde el nodo i hasta el nodo k en la h-ésima iteración del algoritmo está determinada por la ecuación recursiva,

$$ {d_{i,k}}^{h} = \mathrm{Min}\left({d_{ik}}^{h-1}, {d_{ij}}^{h-1} + {d_{jk}}^{h-1}\right) \; i \neq j \neq k \qquad (2.6) $$

Al realizar esta comparación a través de todos los nodos de la red, se obtienen las distancias más cortas entre todos los pares de nodos. La versión en pseudocódigo de este algoritmo, es la siguiente:

$d \leftarrow$ Matriz de distancias
$n \leftarrow$ Número de nodos
Para $j \leftarrow 1$ hasta n
    Para $i \leftarrow 1$ hasta n
        Para $k \leftarrow 1$ hasta n
            Si ($d(i,j)<\infty$) y $d(j,k)<\infty$)
                Si ($i \neq k$ y $j \neq k$ y $i \neq j$)
                    $d(i,k) \leftarrow$ mínimo[$d(i,j)+d(j,k)$, $d(i,k)$];
                Fin
            Fin
        Fin
    Fin
Fin

Si al terminar las iteraciones alguna de las distancias más cortas d_{ik} aún tiene un valor infinito, se concluye que la red no está conectada.

- **Ejemplo numérico**

Dado que únicamente interesa determinar si existe conectividad, se puede suponer que a cada arco corresponde una distancia de 1. Por lo tanto, la matriz de distancias del grafo mostrado en la Figura 2, es

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & \infty & \infty \\ 1 & \infty & 0 & 1 \\ 1 & \infty & 1 & 0 \end{bmatrix}$$

Al aplicar el algoritmo de Floyd, se llega a la siguiente matriz de distancias,

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 2 & 2 \\ 1 & 2 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{bmatrix}$$

Puesto que ninguna de las distancias tiene valor infinito, se concluye que la red está conectada.

2.3.2 Existencia de un árbol de expansión

Se llama **árbol** a un grafo conectado, sin ciclos (Deo, 1974, p. 39), es decir, sin rutas que comienzan y terminan en un mismo nodo. Un árbol es un **árbol de expansión** si contiene el conjunto de todos los vértices (nodos) del grafo (Deo, 1974, p. 55). Por lo tanto, una red conectada es aquella red que tiene por lo menos un árbol de expansión.

Para determinar la existencia de un árbol de expansión, se puede utilizar la siguiente modificación del algoritmo de Prim para el árbol de expansión mínima. Inicialmente se selecciona cualquiera de las

posibles conexiones, y se conecta a los nodos que corresponden a esta conexión. Luego, de manera iterativa se selecciona cualquiera de las posibles conexiones entre los nodos conectados, y los nodos no conectados, y se conectan los nodos respectivos. Este proceso se repite hasta que se cumpla una de las siguientes condiciones: 1) todos los nodos están conectados, 2) hay nodos no conectados y no hay conexiones posibles entre los nodos conectados y los no conectados, lo cual indica que la red no está conectada.

Para la implementación de este algoritmo en Matlab se desarrolló la siguiente formulación basada en la **matriz de adyacencia** del grafo, la cual tiene valores de uno en las posiciones donde el nodo i está conectado directamente con el nodo j, y cero en las demás posiciones:

- Inicialización de vectores
  - Matriz de adyacencia de la red: A
  - Conjunto de nodos de la red: $V \leftarrow \{1, 2, \dots, n\}$
  - Conjunto de nodos conectados: $V_c \leftarrow \emptyset$
  - Conjunto de nodos no conectados : $V_u \leftarrow V$
- Proceso iterativo
  1. Obtenga la matriz reducida R.
     - En la primera iteración $R \leftarrow A$. En las siguientes iteraciones R está conformada por las filas de los nodos no conectados y las columnas de los nodos conectados, es decir, $R \leftarrow A(V_u, V_c)$.
  2. Seleccione un arco de la red
     - En la matriz reducida R localice la posición (h, i) de un elemento diferente de cero. Si en esta matriz no existe ningún elemento diferente de cero, se concluye que la red no está conectada y se termina el algoritmo.
     - En la matriz A determine los índices originales (j, k) del elemento seleccionado en la matriz R.
  3. Actualice los elementos de los conjuntos: nodos conectados V_c y nodos no conectados V_u
     - En la iteración inicial se asigna al conjunto de nodos conectados los índices de la fila y la columna del nodo seleccionado, es decir, $V_c \leftarrow \{j, k\}$. En las sucesivas iteraciones, $V_c \leftarrow V_c \cup j$, es decir, al conjunto de nodos conectados se adiciona el índice de la fila del arco seleccionado en la iteración respectiva.
     - Asigne al conjunto de nodos no conectados la diferencia entre el conjunto de nodos de la red y el conjunto de nodos conectados, es decir, $V_u \leftarrow V - V_c$.

- Si $V_u = \emptyset$, se concluye que la red está conectada y se termina el algoritmo. De lo contrario, se regresa al paso 1.

- **Ejemplo numérico**

Se parte de la matriz de adyacencia de la red de la Figura 2,

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

- Inicialización de vectores

$$V = V_u = \{1, 2, 3, 4\}$$
$$V_c = \emptyset$$

- Primera iteración
  1. En esta primera iteración, $R \leftarrow A$.
  2. Seleccione un arco de la red.
     - Se selecciona el elemento de la posición $(2,1)$ de la matriz R.
     - Dado que en esta primera iteración R es igual a la matriz A, los índices en A son $j = 2$ y $k = 1$.
  3. Actualice los elementos de los conjuntos
     - $V_c \leftarrow \{1, 2\}$
     - $V_u \leftarrow V - V_c$. Entonces $V_u = \{3, 4\}$.
- Segunda iteración
  1. Se obtiene la nueva matriz reducida, $R \leftarrow A(V_u, V_c) = A(\{3, 4\}, \{1, 2\})$.

$$R = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

  2. Seleccione un arco de la red
     - El primer elemento diferente de cero está en la posición (1,1), el cual corresponde a la posición (3, 1) en la matriz A. Por lo tanto, $j = 3$ y $k = 1$.
  3. Actualice los elementos de los conjuntos
     - $V_c \leftarrow \{1, 2\} \cup \{3\} = \{1, 2, 3\}$
     - $V_u \leftarrow V - \{1, 2, 3\} = \{4\}$
- Tercera iteración
  1. La nueva matriz reducida es $R \leftarrow A(\{4\}, \{1, 2, 3\}) = [1 \quad 0 \quad 1]$.
  2. Seleccione un arco de la red.

- El primer elemento diferente de cero está en la posición (1,1), la cual corresponde a la posición (4, 1) de la matriz A. Es decir, $j = 4$ y $k = 1$.

3. Actualice los elementos de los conjuntos
   - $V_c \leftarrow \{1, 2, 3\} \cup \{4\} = \{1, 2, 3, 4\}$
   - $V_u \leftarrow V - \{1, 2, 3, 4\} = \emptyset$
   - Dado que $V_u = \emptyset$, se concluye que la red está conectada, y se termina el algoritmo.

A partir de las conexiones seleccionadas en el algoritmo, (2,1), (3,1) y (4,1), se obtiene el siguiente árbol de expansión de la Figura 4.

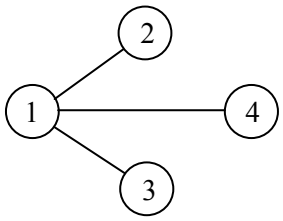

Figura 4. Árbol de expansión de la red

2.3.3 Conectividad algebraica

La matrix de Kirchhoff, también conocida como matriz Laplaciana (Spielman, 2007) es aquella en la cual sus elementos están definidos de la siguiente manera (Deo, 1974):

$$L_{(i,j)} = \begin{cases} -1 \text{ si existe un arco entre el nodo } i \text{ y el nodo } j, & i \neq j, \quad i, j = 1, \dots, n \\ g_i \quad \text{si} \quad i = j \\ 0 \text{ en cualquier otro caso} \end{cases} \tag{2.7}$$

Donde g_i es el grado del nodo i.

Esta matriz también puede obtenerse mediante la diferencia entre una matriz con el grado de los nodos en la diagonal principal, y la matriz de adyacencia.

Fiedler (1973) determinó que la conectividad de una red está relacionada con el segundo menor valor propio de la matriz Laplaciana. Este valor propio es una medida del grado de conectividad del grafo, y por lo tanto se le conoce como **conectividad algebraica** (De Abreu, 2007). La red está conectada sí y sólo si este valor propio es diferente de cero (Fiedler, 1973). De Abreu (2007) presenta una revisión de la literatura sobre conectividad algebraica y sus ámbitos de aplicación.

La implementación de este método comprende cinco pasos: 1) obtención de la matriz Laplaciana, 2) cálculo de los valores propios de dicha matriz, 3) ordenamiento de los valores propios obtenidos, 4) determinación del segundo menor valor propio, y 5) evaluación de la condición de que dicho valor propio sea diferente de cero. Si se cumple esta condición, la red está conectada.

- **Ejemplo numérico**

Nuevamente se parte de la matriz de adyacencia de la red de la Figura 2.

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Al multiplicar esta matriz por - 1 y asignar a las posiciones de la diagonal principal el valor absoluto de la suma de cada fila, se obtiene la matriz Laplaciana,

$$\begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

Los valores propios de esta matriz, son 0, 1, 3 y 4. Puesto que el segundo menor valor propio es mayor que 0, se verifica nuevamente que la matriz está conectada.

2.3.4 Número de árboles de expansión

El número de árboles se expansión se calcula mediante el teorema *Matrix-Tree* de Kirchhoff. De acuerdo con este teorema, el número de árboles de expansión de una red no dirigida es igual a cualquiera de los menores principales de la matriz Laplaciana (Abdesselam, 2003). Este teorema fue enunciado originalmente por Gustav Kirchhoff en 1847.

- **Ejemplo numérico**

Se parte de la matriz Laplaciana obtenida en el ejemplo anterior. El primer menor principal de esta matriz es,

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

El determinante de esta matriz es 3, lo cual indica que la red original tiene 3 árboles de expansión. Por lo tanto, la red está conectada. Los 3 árboles de expansión que se pueden formar a partir de esta red se muestran en la Figura 5.

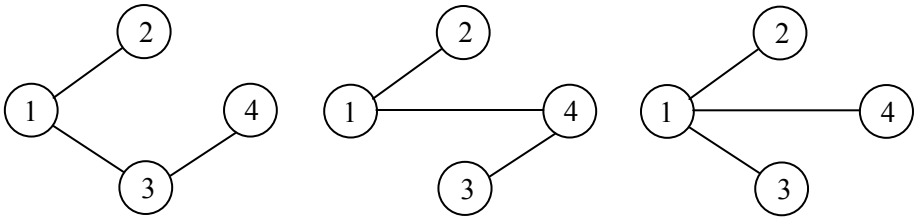

Figura 5. Árboles de expansión de la red

2.3.5 Producto booleano de la matriz de conexión

La matriz de conexión C tiene valores de uno tanto en la diagonal principal como en las entradas donde el vértice i se conecta con el vértice j. Las demás entradas de esta matriz tienen valor de cero. Billinton & Allan (1992, pp. 119-120) describen un método para determinar las rutas mínimas entre todos los nodos de la red, a partir de la reiterativa multiplicación booleana de la matriz de conexión por sí misma, hasta que permanezca inalterado el resultado. Si al finalizar el algoritmo alguna de las entradas por fuera de la diagonal principal de la matriz resultante es igual a cero, entonces no existe una ruta desde el nodo i hasta el nodo j, y por lo tanto, la matriz no está conectada.

- **Ejemplo numérico**

Para este ejemplo se adopta la siguiente notación: $\boldsymbol{C^l}$ es la $\boldsymbol{l}$-ésima potencia de la matriz de conexión $\boldsymbol{C}$, es decir, $\boldsymbol{C^l}$ es el producto matricial de $\boldsymbol{l}$ copias de la matriz de conexión. $\boldsymbol{C^{(l)}}$ es el $\boldsymbol{l}$-ésimo producto booleano de la matriz de conexión $\boldsymbol{C}$.

Se parte de la matriz de conexión C. Se hace $C = C^{(1)}$:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

La potencia 2 de esta matriz es:

$$\begin{bmatrix} 4 & 2 & 3 & 3 \\ 2 & 2 & 1 & 1 \\ 3 & 1 & 3 & 3 \\ 3 & 1 & 3 & 3 \end{bmatrix}$$

En las operaciones booleanas los únicos valores posibles de las variables son uno o cero: uno cuando el valor de la entrada es mayor o igual que uno; cero en caso contrario. De esta manera, la matriz booleana que corresponde al anterior resultado es una matriz de unos de dimensión 4**.**

Puesto que ya todas las celdas tienen el valor de 1, no es necesario volver a multiplicar la matriz. La matriz resultante indica que todos los nodos de la red están conectados. El valor de cero en alguna de las entradas o posiciones de la matriz indicaría lo contrario.

2.3.6 Potencia de la matriz de conexión

Este método es una variante del método anterior. Debido a que como máximo hay $n - 1$ arcos entre cualquier par de nodos de la red, el método consiste en obtener directamente la potencia $n - 1$ de la matriz C. Si alguna de las entradas por fuera de la diagonal principal de la matriz resultante es igual a cero, la red no está conectada.

- **Ejemplo numérico**

Dado que la red tiene 4 nodos, se calcula la potencia 3 de la matriz de conexión,

$$C^3 = \begin{bmatrix} 12 & 6 & 10 & 10 \\ 6 & 4 & 4 & 4 \\ 10 & 4 & 9 & 9 \\ 10 & 4 & 9 & 9 \end{bmatrix}$$

Puesto que ninguna de las entradas por fuera de la diagonal principal es diferente de cero, la red está conectada.

2.3.7 Multiplicación de la matriz de adyacencia

El número de rutas de longitud l, desde el vértice i hasta el vértice j, corresponde a la entrada (i, j) de la potencia l de la matriz de adyacencia A (Biggs, 1974, p. 9). De esta propiedad se deriva que si alguna de las entradas (i, j), por fuera de la diagonal principal de la matriz

$$B = \sum_{i=1}^{n-1} A^l \tag{2.8}$$

es igual a cero, entonces no hay conexión entre los nodos j e i, y por lo tanto el grafo asociado a la matriz de adyacencia A no está conectado. Este método puede adaptarse fácilmente a situaciones en donde además de la conectividad se requiera establecer condiciones o criterios con respecto a la distancia entre los nodos o el diámetro de la red.

2.4 Comparación de tiempos de ejecución

La Figura 6 muestra los tiempos de ejecución de los métodos descritos en las Secciones 2.3.1 - 2.3.6, considerando diferentes tamaños de red. Se excluyó de esta comparación el método de la multiplicación de la matriz de adyacencia (Sección 2.3.7), el cual se presume menos eficiente debido a que requiere el cálculo de varias potencias de una matriz. Para cada tamaño de red se generaron aleatoriamente 50 redes no dirigidas con densidad promedio 0.5. El código para la comparación en Matlab se incluye en el Apéndice 1.

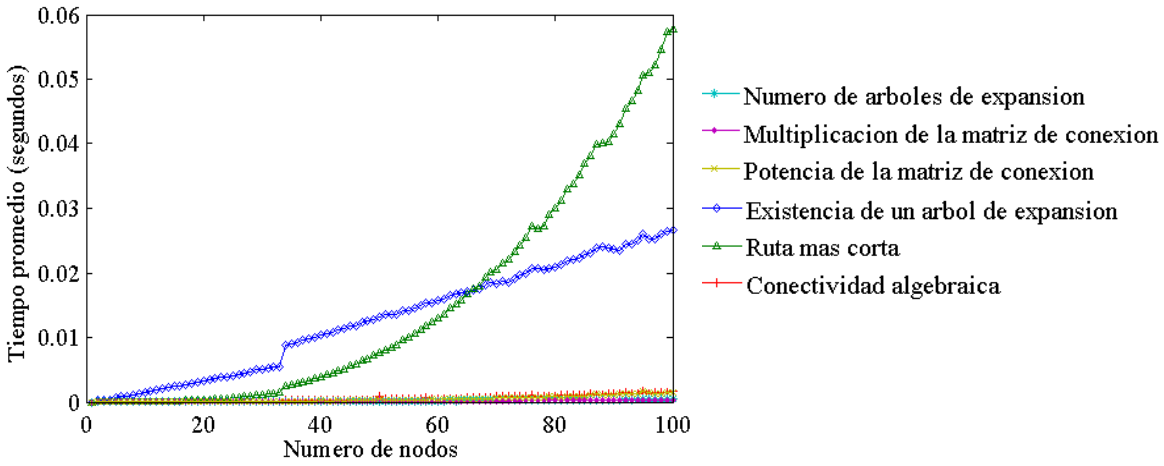

Figura 6. Tiempos de ejecución de los métodos para determinación de conectividad considerando diferentes tamaños de red (50 réplicas, densidad 0.5)

Como se puede observar en la Figura 6, los métodos *identificación de un árbol de expansión* y *rutas más cortas* tuvieron los más altos tiempos de ejecución. Los tiempos de ejecución de los cuatro métodos restantes fueron comparados en redes con densidad promedio 0.5, con un máximo de 100 nodos, y con cien réplicas para cada tamaño de red. Claramente se puede distinguir que los métodos *número de árboles de expansión* y la *multiplicación booleana de la matriz de conexión* alcanzaron los menores tiempos de ejecución (Figura 7).

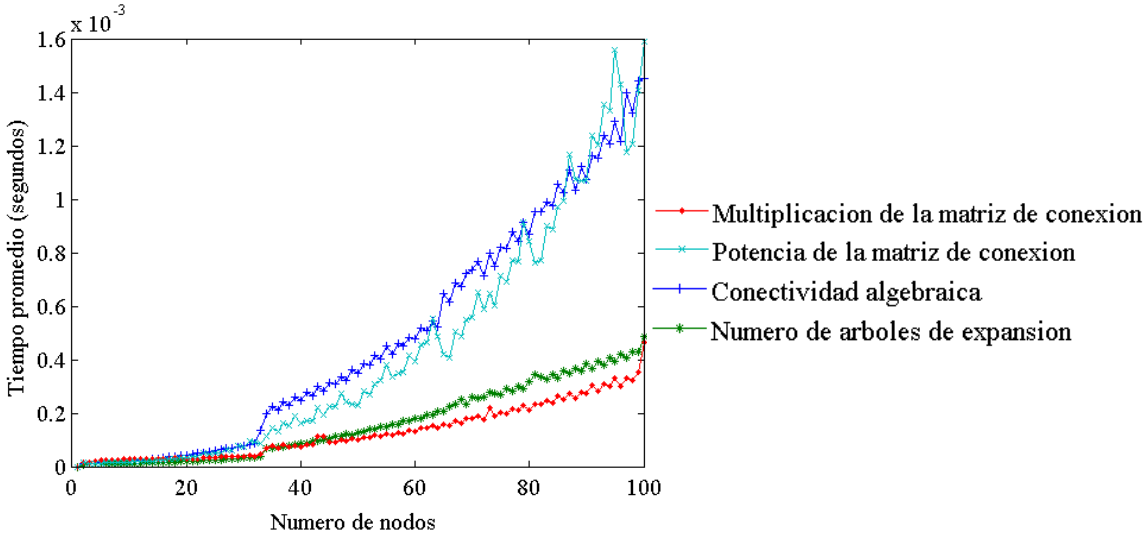

Figura 7. Tiempos de ejecución de cuatro métodos para determinación de conectividad considerando diferentes tamaños de red (100 réplicas).

Para comparar los dos métodos con los menores tiempos de ejecución, se realizó un experimento considerando densidades entre 0.1 y 0.9, con incrementos de 0.1, y tamaños de red entre 3 y 100

nodos. Se obtuvieron 100 réplicas de cada condición experimental. A partir de los datos obtenidos se ajustó un modelo lineal con las siguientes variables:

- Número de nodos de la red:
- Densidad de la red:
- Método de evaluación de conectividad:
-

Como variable de respuesta, se definió la potencia 0.1 del tiempo de ejecución, es decir, . El modelo ajustado fue,

Los resultados del análisis de varianza y la regresión, se muestran en la Tabla 1.

Tabla 1. Análisis de varianza y análisis de regresión de los tiempos de corrida de los algoritmos

Source	SS	df	MS
Model	1.4232147	5	.28464294
Residual	.106486708	983	.000108328
Total	1.52970141	988	.001548281

Number of obs = 989
F(5, 983) = 2627.60
Prob > F = 0.0000
R-squared = 0.9304
Adj R-squared = 0.9300
Root MSE = .01041

Source	Partial SS	df	MS	F	Prob > F
Model	1.4232147	5	.28464294	2627.60	0.0000
densidad	.046161031	1	.046161031	426.12	0.0000
nodos	.00413585	1	.00413585	38.18	0.0000
nodos#nodos	.003395345	1	.003395345	31.34	0.0000
nodos#nodos#nodos	.005109925	1	.005109925	47.17	0.0000
metodo	.05816584	1	.05816584	536.94	0.0000
Residual	.106486708	983	.000108328		
Total	1.52970141	988	.001548281		

tiempo_pow01	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
densidad	-.0241421	.0011695	-20.64	0.000	-.0264372	-.0218471
nodos	.0007992	.0001293	6.18	0.000	.0005454	.0010531
c.nodos# c.nodos	.000016	2.85e-06	5.60	0.000	.0000104	.0000216
c.nodos# c.nodos# c.nodos	-1.24e-07	1.81e-08	-6.87	0.000	-1.59e-07	-8.86e-08
2.metodo	.0153379	.0006619	23.17	0.000	.014039	.0166368
_cons	.3298563	.001731	190.55	0.000	.3264593	.3332532

Aunque los residuales de este análisis no pasaron la prueba de normalidad, el diagrama de probabilidad normal no evidencia una distribución de los residuales muy distante de la distribución normal.

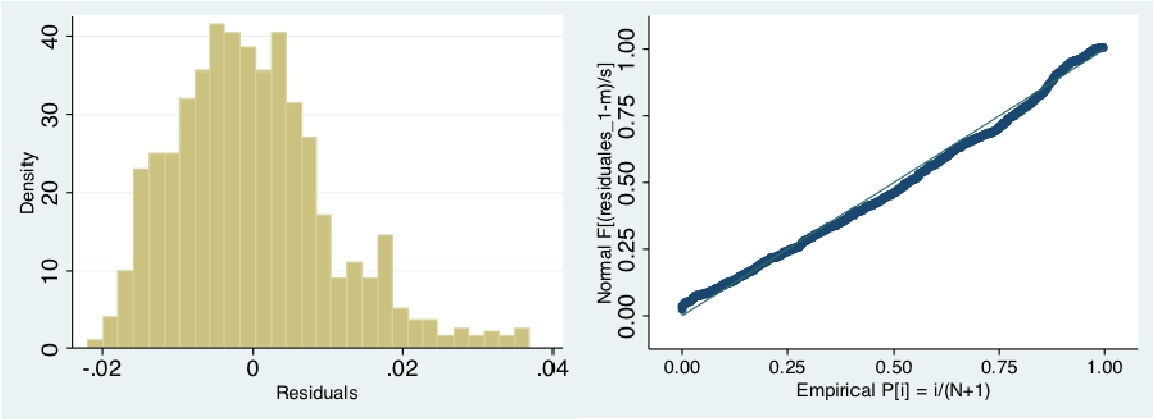

Figura 8. Distribución de los errores del modelo de regresión para los tiempos de ejecución de los algoritmos de evaluación de confiabilidad.

Por lo tanto, de acuerdo con la regresión realizada, el 93% de la variabilidad puede ser explicada mediante el siguiente modelo de regresión:

Dado que en este modelo el valor del parámetro correspondiente a la variable que representa el método 2 tiene valor positivo (0.0153379), se puede concluir que el método del *número de árboles de expansión* tiende a lograr menores tiempos de ejecución, considerando diferentes densidades y tamaños de red. Este método es utilizado en el algoritmo que se presenta en la Sección 2.5.

2.5 Algoritmo general para determinación de conectividad

Para el cálculo del número de árboles de expansión la operación más demandante computacionalmente es el cálculo del determinante de la matriz Laplaciana. En algunas redes se puede evitar esta operación mediante la evaluación de las condiciones necesarias y suficientes discutidas en la Sección 2.2, tal como se muestra en la Figura 9.

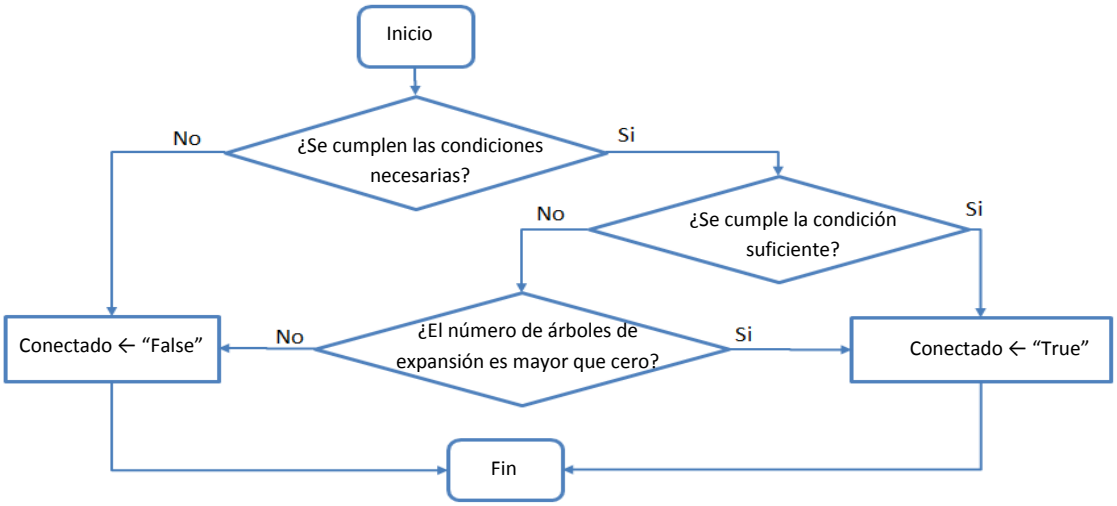

Figura 9. Secuencia de condiciones para determinación de conectividad

Siguiendo la lógica de la Figura 9, el algoritmo general para evaluación de conectividad se muestra en la Figura 10.

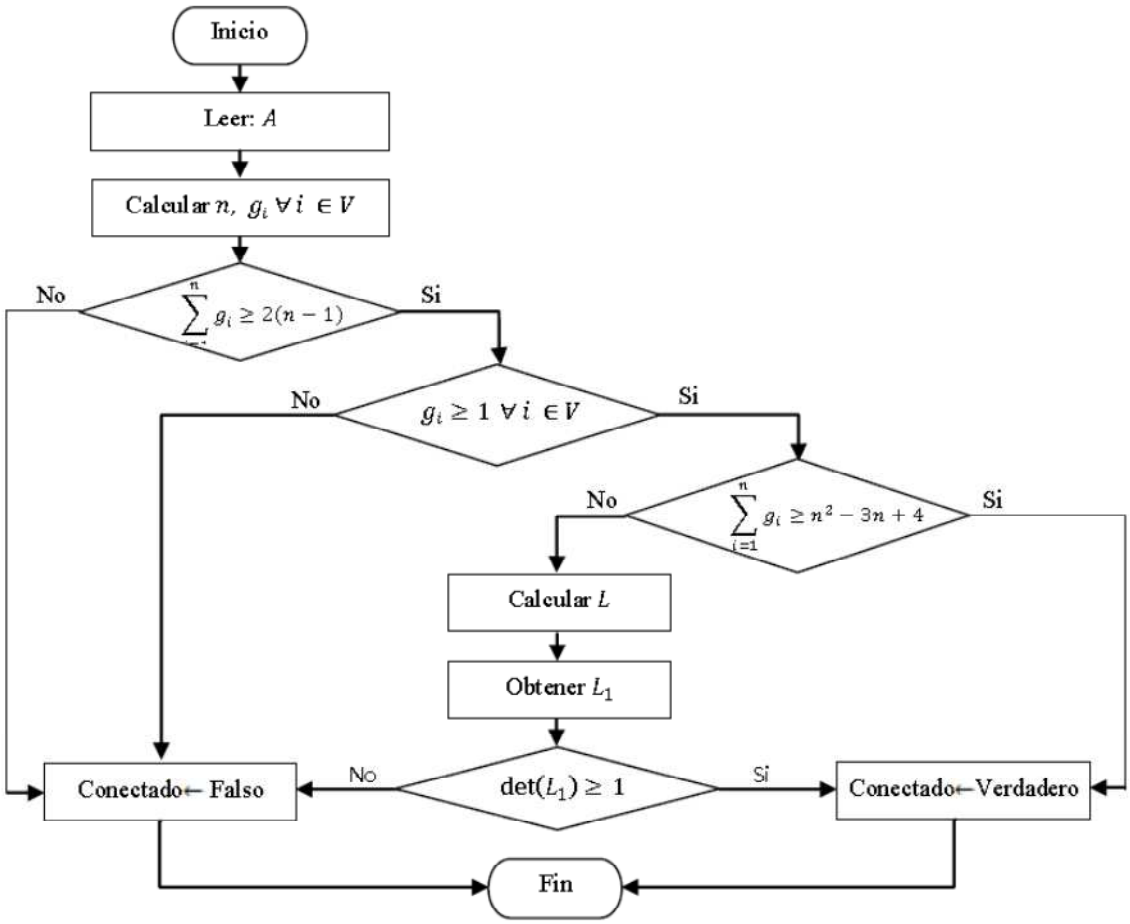

Figura 10. Algoritmo para determinación de conectividad de redes generales no dirigidas

A partir de las condiciones de conectividad y del algoritmo general para evaluación de conectividad, en el capítulo siguiente se proponen métodos para evaluación de confiabilidad.

3 Evaluación de la confiabilidad de redes

En este capítulo se desarrolla un algoritmo para la evaluación de la confiabilidad de redes generales. Para lograr que el método se desempeñe eficientemente en redes de diferentes tamaños, densidades y configuraciones, se combinan métodos de cálculo exacto, técnicas de reducción, simulación Monte Carlo y cotas de confiabilidad.

Inicialmente se presenta una descripción general de los métodos de cálculo exacto y de los métodos Monte Carlo. Luego se deriva el algoritmo propuesto.

3.1 Métodos exactos

Los métodos de cálculo exacto de confiabilidad de redes generales requieren la enumeración de los estados de la red, los cortes mínimos o el listado de árboles de expansión [véase por ejemplo Aggarwal y Rai (1981), Satyanarayana & Hagstrom (1981), Liu *et al* .(1999), Cancela *et al*. (2001)]. En una red con n nodos y m conexiones, el total de posibles estados es 2^m. Si la red está completamente conectada tiene n^{n-2} árboles de expansión y $2^{n-1}-1$ cortes mínimos (Rubino, 1998, pp. 281-282). Por lo tanto, los tiempos de ejecución de estos métodos tienden a crecer exponencialmente, lo cual los hace imprácticos para redes de mediano o gran tamaño. A pesar de esta limitación, en redes pequeñas o de topologías simples, los tiempos de ejecución de estos métodos exactos pueden ser menores que los tiempos requeridos por los métodos Monte Carlo.

Se describen a continuación dos métodos exactos para el cálculo de la confiabilidad: enumeración completa y expansión booleana de la función de estructura. Posteriormente se discute un método para el cálculo de la confiabilidad a partir del listado de árboles de expansión de la red.

3.1.1 Enumeración completa

Cada sistema tiene una función característica denominada **función de estructura** $\phi(\mathrm{s})$, tal que

$$\phi(\mathrm{s}) = \begin{cases} 1, & \text{si el sistema funciona} \\ 0, & \text{en caso contrario} \end{cases}$$

En general, esta función es no trivial, ya que depende de la estructura específica de cada red. En esta función, $\mathrm{s} = [s_1, s_2, \ldots s_i, \ldots s_m]$ es el **vector de estado del sistema**, donde $s_i = \{0,1\}$ representa el estado del *i*-ésimo componente (1 si el componente funciona ó 0 si el componente falla). Para representar la posibilidad de falla de los componentes, cada s_i puede definirse como una variable aleatoria Bernoulli con parámetro p_i, la probabilidad de que el componente funcione.

El valor esperado de la función de estructura $\phi(\mathrm{s})$ corresponde a la confiabilidad del sistema. Es decir,

$$R_s = E[\phi(\mathrm{s})] = \sum_{\forall j} \Pr(\mathrm{s_j})\phi(\mathrm{s_j}) \tag{3.1}$$

Donde,

- $\mathrm{s_j}$: el j-ésimo vector de estado del sistema, para $j = 1, 2, \ldots, 2^m$.
- $\Pr(\mathrm{s_j})$:la probabilidad de cada vector de estado de la red, tal que (Rubino, 279, 1998)

$$\Pr(\mathrm{s_j}) = \left(\prod_{i \mid s_{ij}=1} p_i \right) \left(\prod_{i \mid s_{ij}=0} (1 - p_i) \right) \quad , \forall j \tag{3.2}$$

A pesar del crecimiento exponencial del número de términos de la sumatoria de la Ecuación (3.1), este método tiene la ventaja de no requerir la determinación de cortes o árboles de la red. La codificación en Matlab se muestra en el Apéndice 2.

Ejemplo numérico

Para este ejemplo se utiliza la misma red de la Figura 2. En esta ocasión, al lado de cada conexión se indica la confiabilidad correspondiente (Figura 11).

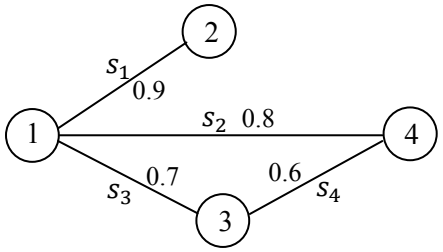

Figura 11. Red no dirigida con valores de confiabilidad en sus conexiones

El cálculo de la confiabilidad de esta red se detalla en la Tabla 2.

Tabla 2. Cálculo de la confiabilidad mediante enumeración completa

Estado j de la red $(\mathbf{s_j})$	s_i				$\phi(\mathrm{s_j})$	p_i si $s_{ij}=1$ / $(1-p_i)$ si $s_{ij}=0$				$\Pr(\mathbf{s_j})$	$\Pr(\mathbf{s_j})\phi(\mathbf{s_j})$
	s_1	s_2	s_3	s_4							
$\mathbf{s_1}$	0	0	0	0	0	-	-	-	-	-	0
$\mathbf{s_2}$	0	0	0	1	0	-	-	-	-	-	0
$\mathbf{s_3}$	0	0	1	0	0	-	-	-	-	-	0
$\mathbf{s_4}$	0	0	1	1	0	-	-	-	-	-	0
$\mathbf{s_5}$	0	1	0	0	0	-	-	-	-	-	0
$\mathbf{s_6}$	0	1	0	1	0	-	-	-	-	-	0
$\mathbf{s_7}$	0	1	1	0	0	-	-	-	-	-	0
$\mathbf{s_8}$	0	1	1	1	0	-	-	-	-	-	0
$\mathbf{s_9}$	1	0	0	0	0	-	-	-	-	-	0
$\mathbf{s_{10}}$	1	0	0	1	0	-	-	-	-	-	0
$\mathbf{s_{11}}$	1	0	1	0	0	-	-	-	-	-	0
$\mathbf{s_{12}}$	1	0	1	1	1 →	0.9	0.2	0.7	0.6	0.0756	0.0756
$\mathbf{s_{13}}$	1	1	0	0	0	-	-	-	-	-	0
$\mathbf{s_{14}}$	1	1	0	1	1 →	0.9	0.8	0.3	0.6	0.1296	0.1296
$\mathbf{s_{15}}$	1	1	1	0	1 →	0.9	0.8	0.7	0.4	0.2016	0.2016
$\mathbf{s_{16}}$	1	1	1	1	1 →	0.9	0.8	0.7	0.6	0.3024	0.3024
										$\sum \Pr(\mathbf{s_j})\phi(\mathbf{s_j}) =$	0.7092

3.1.2 Expansión booleana de la función de estructura

Mientras que en redes de dos terminales la función de estructura puede obtenerse a partir de las rutas mínimas (rutas que no visitan cada nodo más de una vez) entre los nodos origen y destino, en las redes de conectividad global la función de estructura puede construirse a partir de los árboles de expansión de la red, en lugar de las rutas mínimas (Rubino, 1998).

En la red utilizada como ejemplo (Figura 11), los árboles de expansión son $\{s_1, s_3, s_4\}$, $\{s_1, s_2, s_4\}$ y $\{s_1, s_2, s_3\}$. De esta manera, la función de estructura correspondiente se puede definir como un arreglo en paralelo de los árboles de expansión de la red,

$$\phi(\mathrm{s}) = 1 - (1 - s_1 s_3 s_4)(1 - s_1 s_2 s_4)(1 - s_1 s_2 s_3)$$
$$\phi(\mathrm{s}) = 1 - (1 - s_1 s_3 s_4 - s_1 s_2 s_4 - s_1 s_2 s_3 + 2 s_1 s_2 s_3 s_4)$$
$$= s_1 s_3 s_4 + s_1 s_2 s_4 + s_1 s_2 s_3 - 2 s_1 s_2 s_3 s_4$$

Al reemplazar las variables binarias s_i por las confiabilidades p_i correspondientes a cada arco, se obtiene la función de confiabilidad de la red,

$$R_s = p_1 p_3 p_4 + p_1 p_2 p_4 + p_1 p_2 p_3 - 2 p_1 p_2 p_3 p_4$$

Sustituyendo los valores mostrados en la Figura 11, $p_1 = 0.9,\ p_2 = 0.8,\ p_3 = 0.7,\ p_4 = 0.6$, se obtiene una confiabilidad de $R_s = 0.7092$, el cual coincide con el resultado anteriormente obtenido. Este método es inviable en redes de mayor tamaño, no sólo por el incremento exponencial del número de árboles de expansión, sino también por el esfuerzo computacional para obtener la expansión booleana de la función de confiabilidad.

3.1.3 Cálculo mediante el principio de inclusión-exclusión

Sea P_{T_j} la probabilidad de que el j-ésimo árbol de expansión esté funcionando. La confiabilidad global de la red se puede calcular como la unión de las probabilidades P_{T_j}(Rubino, 1998). Debido a que estas probabilidades no corresponden a eventos disjuntos, se debe aplicar el principio de inclusión-exclusión, también conocido como teorema de Poincaré y Sylvester [(Rubino, 1998), (Kuo & Zuo, 2003)],

$$R_s = \sum_{m=1}^{q} (-1)^{m-1} \sum_{1 \le T_1 < T_2 < \cdots < T_m \le q} \Pr\left(\bigcap_{j=1}^{m} P_{T_j} \right) \tag{3.3}$$

Donde q es el total de árboles de expansión de la red.

Este teorema se puede escribir más explícitamente como,

$$R_s = \sum_{i=1}^{q} \left(\prod_{h \in T_i} r_h \right) - \sum_{i,j:1 \le i < j \le q} \left(\prod_{h \in T_i \cup T_j} r_h \right) + \sum_{i,j,k:1 \le i < j < k \le q} \left(\prod_{h \in T_i \cup T_j \cup T_k} r_h \right) - \cdots$$
$$+ (-1)^{q-1} \left(\prod_{h \in T_1 \cup T_2 \cup \ldots \cup T_q} r_h \right) \tag{3.4}$$

Donde T_i es el conjunto de los arcos que componen el i-ésimo árbol de expansión y r_h es la confiabilidad del arco h. La implementación en Matlab se muestra en el Apéndice 4.

Para generar el listado de árboles de expansión se ha escogido el método discutido por Aggarwal & Rai (1981), dada su facilidad de implementación. Este método utiliza el producto cartesiano de $n-1$ cortes mínimos de la red. Recuérdese que el producto cartesiano de dos conjuntos A, B está definido como $A \times B = \{(x, y) | (x \in A) \wedge (y \in B)\}$.

Matricialmente, la forma más práctica de obtener este subconjunto de cortes mínimos consiste en definir cada corte como una de las filas de la matriz de adyacencia. Al operar el producto cartesiano pueden obtenerse circuitos, los cuales se caracterizan por tener un número par de apariciones (Aggarwal & Rai, 1981). Estos circuitos deben ser identificados y removidos. También se deben suprimir los elementos que tienen menos de $n-1$ arcos, ya que estos no tienen un número suficiente

de arcos para conformar un árbol de expansión. Después de realizar estas operaciones, el conjunto resultante está conformado por el listado de árboles de expansión de la red. El programa correspondiente en Matlab se incluye en el Apéndice 5.

Ejemplo numérico

La matriz de adyacencia de la red de la Figura 11 es,

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

Ahora, se reemplazan los valores de 1 por las variables binarias correspondientes.

$$A = \begin{bmatrix} 0 & s_1 & s_3 & s_2 \\ s_1 & 0 & 0 & 0 \\ s_3 & 0 & 0 & s_4 \\ s_2 & 0 & s_4 & 0 \end{bmatrix}$$

En la Sección 2.3.4 se determinó que los árboles de expansión de esta red son $s_1s_3s_4$, $s_1s_2s_4$ y $s_1s_2s_3$. Para corroborarlo, se hace uso del método basado en el producto cartesiano de $n-1$ cortes mínimos de la red. Se seleccionan como cortes las primeras tres filas de la matriz A: (s_1, s_2, s_3), (s_1) y (s_3, s_4). A continuación, se obtiene el producto cartesiano de estos conjuntos,

$$(s_1, s_2, s_3) \times (s_1) \times (s_3, s_4) = (s_1, s_1s_2, s_1s_3) \times (s_3, s_4)$$
$$= (s_1s_3, s_1s_2s_3, s_1s_3, s_1s_4, s_1s_2s_4, s_1s_3s_4)$$

Al suprimir los elementos que tienen menos de 3 arcos, ya que no conforman un árbol de expansión, se obtiene

$$= (s_1s_2s_3, s_1s_2s_4, s_1s_3s_4)$$

En este resultado no hay circuitos ya que ninguno de los elementos tiene un número par de apariciones. Por consiguiente, se han obtenido los árboles de expansión de la red.

Expandiendo la Ecuación (3.4) para los tres árboles de expansión de la red, se obtiene

$$R_s = \sum_{i=1}^{3} \left(\prod_{h \in T_i} p_h \right) - \prod_{h \in T_1 \cup T_2} p_h - \prod_{h \in T_1 \cup T_3} p_h - \prod_{h \in T_2 \cup T_3} p_h + \prod_{h \in T_1 \cup T_2 \cup T_3} p_h$$

Para evitar la manipulación simbólica de variables, los árboles de expansión se expresan mediante vectores binarios, en los que las posiciones con valores de uno indican la existencia de arcos en las conexiones correspondientes,

$$T_1 = s_1s_2s_3 = [1, 1, 1, 0]$$

$$T_2 = s_1 s_2 s_4 = [1, 1, 0, 1]$$
$$T_3 = s_1 s_3 s_4 = [1, 0, 1, 1]$$

A continuación se obtienen las uniones de los conjuntos. En este caso, el resultado es el mismo en todas las uniones,

$$T_1 \cup T_2 = T_1 \cup T_3 = T_2 \cup T_3 = [1, 1, 1, 1]$$

También,

$$T_1 \cup T_2 \cup T_3 = [1, 1, 1, 1]$$

Ahora, se obtiene el producto de las confiabilidades correspondientes a los elementos de cada uno de los vectores binarios obtenidos anteriormente,

$$P(T_1 = \text{conectado}) = \prod_{h \in T_1} p_h = p_1 p_2 p_3 = 0.504$$
$$P(T_2 = \text{conectado}) = \prod_{h \in T_2} p_h = p_1 p_2 p_4 = 0.432$$
$$P(T_3 = \text{conectado}) = \prod_{h \in T_3} p_h = p_1 p_3 p_4 = 0.378$$

Similarmente,

$$P((T_1 \cup T_2) = \text{conectado}) = \prod_{h \in T_1 \cup T_2} p_h = 0.3024$$

Este mismo resultado corresponde a la productoria de las uniones $T_1 \cup T_3$, $T_2 \cup T_3$ y $T_1 \cup T_2 \cup T_3$. De esta manera la Ecuación (3.4), se reduce a,

$$R_s = 0.504 + 0.432 + 0.378 - (3 \times 0.3024) + 0.3024$$
$$R_s = 0.7092$$

Nuevamente se ha obtenido el mismo resultado para la red de la Figura 11. En redes con muy pocos árboles de expansión este método alcanzaría tiempos de ejecución competitivos con métodos de evaluación aproximada de confiabilidad, los cuales se explican a continuación.

3.2 Métodos Monte Carlo

Tres motivos justifican el uso de métodos aproximados para calcular la confiabilidad. En primer lugar, permitirían estimar la confiabilidad en tiempo polinomial (y no exponencial, como ocurre con los métodos exactos). Segundo, la confiabilidad exacta de los sistemas reales es muy difícil o imposible de calcular, debido a inexactitudes de los datos y dependencias entre eventos. Tercero, en aplicaciones prácticas un valor aproximado de la confiabilidad puede ser aceptable (Colbourn, 1999 , p. 138).

En vista de lo anterior, en este trabajo se considera el uso de métodos Monte Carlo, los cuales permiten estimar aproximadamente la confiabilidad en tiempos polinomiales, y pueden utilizarse de manera flexible con cualquier topología de red. A continuación se describen algunos conceptos básicos de los métodos Monte Carlo.

3.2.1 Varianza del estimador

El enfoque crudo de simulación Monte Carlo consiste en calcular el valor promedio de la función de estructura de la red, a partir de una muestra aleatoria de vectores de estado de los componentes. En este caso, la varianza del estimador está determinada por el tamaño N de la muestra y por la confiabilidad real R_s de la red (Rubino, 1998, p. 287),

$$V(\hat{R}_s) = \frac{R_s(1 - R_s)}{N} \tag{3.5}$$

El principal reto en la implementación de métodos Monte Carlo, consiste en obtener una estimación con precisión adecuada, en un tiempo razonable. Para disminuir el número de iteraciones necesarias para la estimación de la confiabilidad, se utilizan técnicas que permitan disminuir la varianza del estimador.

Las técnicas de reducción de varianza parten del principio de que la simulación es un experimento estadístico controlado, y que como tal, se puede reducir la varianza de un cierto número de réplicas, mediante la aplicación de principios probabilísticos (Lewis & Orav, 1989, p. 334). Existen siete técnicas clásicas de reducción de varianza (Lewis & Orav, 1989): variables antitéticas, variables de control, variables de control con regresión, muestreo sistemático, muestreo condicional, muestreo de importancia y estratificación. Rubino (1998) describe algunos de los métodos de reducción de varianza que se han aplicado para la estimación de la confiabilidad de redes. Algunos de dichos métodos requieren información complementaria, tal como listas de árboles o cortes mínimos, lo cual incrementa el esfuerzo computacional y en ocasiones puede incrementar los tiempos de ejecución (Rubino, 1998, p. 288).

La eficiencia es una medida de la calidad de los estimadores y se constituye en el principal criterio a la hora de seleccionar la técnica de reducción de varianza a utilizar. De acuerdo con Lemieux (2009, p. 89), la eficiencia se define como

$$\text{Eff}(\hat{\mu}) = \left([Var(\hat{\mu}) + B^2(\hat{\mu})] \cdot C(\hat{\mu})\right)^{-1} \tag{3.6}$$

Donde,

$B^2(\hat{\mu})$: Es el sesgo del estimador, es decir $E(\hat{\mu}) - \mu$

$C(\hat{\mu})$: Es el tiempo esperado de cómputo para $\hat{\hat{\mu}}$

De acuerdo con este criterio, se considera que la técnica a utilizar es la de variables antitéticas. Como se discute más adelante, esta técnica: 1) asegura la reducción de varianza en la estimación de confiabilidad de sistemas coherentes, 2) requiere sólo la mitad de números aleatorios que utiliza el método de simulación cruda; por lo tanto, demanda un menor esfuerzo computacional y un menor tiempo ejecución (Lemieux, 2009, p. 100).

La técnica de variables antitéticas busca generar pares de observaciones con una alta correlación negativa, lo cual conlleva una disminución de la varianza total. En general, la varianza resultante es igual a (Lemieux, 2009, p. 91),

$$V = \frac{\sigma^2}{n} + \frac{1}{n}\mathrm{Cov}\big(f(\boldsymbol{U}), f(\overline{\boldsymbol{U}})\big) \tag{3.7}$$

Donde, $\boldsymbol{U} = (U_1, \dots, U_s) \sim U([0,1))^s$, y $\overline{\boldsymbol{U}} = (1 - U_1, \dots, 1 - U_s)$.

De esta expresión se deduce que la varianza del método de variables antitéticas es menor que la varianza del método Monte Carlo crudo, siempre que la correlación entre $f(\boldsymbol{U})$ y $f(\overline{\boldsymbol{U}})$ sea negativa. Cuanta más negativa es la correlación entre pares de observaciones, mayor es la efectividad del método de variables antitéticas. En problemas de cálculo de confiabilidad de redes generales, el término de covarianzas y por consiguiente, la varianza resultante, pueden ser difíciles de calcular. No obstante, el siguiente teorema permite determinar en qué casos la aplicación de esta técnica asegura una disminución de la varianza (Lemieux, 2009, p. 94),

> ***Teorema:*** *Sea* $f: [0,1)^s \longrightarrow R$ *una función acotada y monótona en cada uno de sus argumentos. Supóngase además que la función no es constante en el interior de su dominio. Entonces,* $Cov\big(f(\boldsymbol{U}), f(\overline{\boldsymbol{U}})\big) < 0$.

Los sistemas coherentes se caracterizan por tener una función de estructura no decreciente en cada uno de sus argumentos (Kuo *et al.*, 2001, p. 21); por lo tanto, son monótonos. De esta manera, el teorema anterior garantiza que en sistemas coherentes la covarianza es negativa, lo cual implica que la varianza del método de variables antitéticas es inferior a la varianza del método crudo de simulación Monte Carlo.

El estimador de la varianza de la confiabilidad, correspondiente a N iteraciones de la simulación con variables antitéticas, está dado por (El Khadiri & Rubino, 1992, p. 9),

$$\hat{V} = \frac{V_N}{N(N-1)} + \frac{\hat{R}_s^{\,2}}{N-1} \tag{3.8}$$

Tal que,

$$\hat{R}_s = \frac{1}{2N}\sum_{k=1}^{N} Q_k$$

$$V_N = \sum_{k=1}^{N}\left(\frac{Q_k}{2}\right)^2$$

$$Q_k = \phi(\bar{\boldsymbol{U}}_{\boldsymbol{k}}) + \phi(\boldsymbol{U}_{\boldsymbol{k}})$$

$\boldsymbol{U}_{\boldsymbol{k}}$ es el vector de números aleatorios $[U_{1k}, U_{2k}, \dots, U_{mk}]$ en la k-ésima iteración del algoritmo, y $\phi(\cdot)$ es la función de estructura de la red.

3.2.2 Algoritmo Monte Carlo con variables antitéticas para la evaluación de confiabilidad de redes generales

Sea M la matriz de confiabilidades de la red, definida de la siguiente manera

$$M_{(i,j)} = \begin{cases} p_{ij} & \text{si hay una conexión entre los nodos } j \text{ e } i\,, \quad i \neq j \\ 0 & \text{en caso contrario} \end{cases}$$

A partir de esta matriz se puede estimar la confiabilidad de la red, mediante el siguiente algoritmo.

```
Inicio
N ← Número de réplicas
M ← Matriz de confiabilidades de la red
n ← Número de nodos de la red (número de filas de M)
v ← Vector con elementos de la matriz triangular superior de M
m ← Número de elementos del vector v
Q_k ← 0
Para i ← 1 hasta N
    Generar U %% Vector de números aleatorios de 1 × m
    C1 ← ceros(n, n)
    Para j ← 1 hasta m
          Si U1(j) < v(j)
                a(j) ← 1
          De lo contrario
                a(j) ← 0
          Fin
    Fin
    tris(C1,1) ← a
    C1 = C1+ C1′
```

```
    C2 ← ceros(n, n)
    U2 ← (unos(1, m)- U)
    Para j ← 1 hasta m
          Si U2(j) < v(j)
                a(j) ← 1
          De lo contrario
                a(j) ← 0
          Fin
    Fin
    tris(C2,1) ← a
      C2 = C2 + C2'
    Q_k = (conectividad(C1)+conectividad(C2))/2
    P = Q_k+P
    V = V +Q_k^2
Fin
R_s ← p/N
Var = V/( N*( N − 1)) + R_s
Fin
```

Como respuesta se obtienen la confiabilidad de la red R_s y la varianza Var.

Para este algoritmo, se definen las siguientes funciones:

`tris`(C, k): Matriz triangular superior de C, a partir de la k-ésima diagonal

`conectividad`(C): Función que determina si existe conectividad en la red correspondiente la matriz C. Retorna 1 si existe conectividad, y 0 en caso contrario.

`unos`(r, s): Matriz de unos de dimensión $r \times s$

`ceros`(r, s): Matriz de unos de dimensión $r \times s$

El código en Matlab correspondiente a este algoritmo se muestra en el Apéndice 6.

3.3 Cotas de confiabilidad

Como su nombre lo indica, las cotas ofrecen estimaciones de los límites superior e inferior del rango en el cual yace la confiabilidad de un sistema. Aunque dichas estimaciones no son muy ajustadas, típicamente los cálculos se pueden realizar en tiempos significativamente menores que con los métodos exactos o los métodos Monte Carlo.

Algunos de los métodos desarrollados para el cálculo de cotas requieren del conjunto de cortes mínimos de la red, lo cual es en sí mismo un problema NP-Hard (Rubino, 1998). Jan (1993) propuso un método para el cálculo de la cota superior de la confiabilidad global. Este método solamente requiere los cortes mínimos que separan nodos individuales de la red, lo cual puede ser determinado en

tiempo polinomial (Sharafat & Ma'rouzi, 2009). Utilizando un enfoque similar Konak & Smith (1998) desarrollaron una cota superior más ajustada, la cual permite considerar que los arcos de la red pueden tener diferentes confiabilidades. La fórmula propuesta por Konak & Smith (1998) es,

Donde,

: Conjunto de arcos conectados al nodo

: Confiabilidad del arco no dirigido

Cuando la red no está conectada, el lado derecho de la inecuación 3.9 puede tomar valores negativos. En 50 redes generadas aleatoriamente se obtuvieron las estimaciones de la confiabilidad mediante simulación Monte Carlo y se calculó la cota superior (ver Figura 12). En el Apéndice 7 se incluye la implementación de la función para el cálculo de la cota superior de confiabilidad en Matlab.

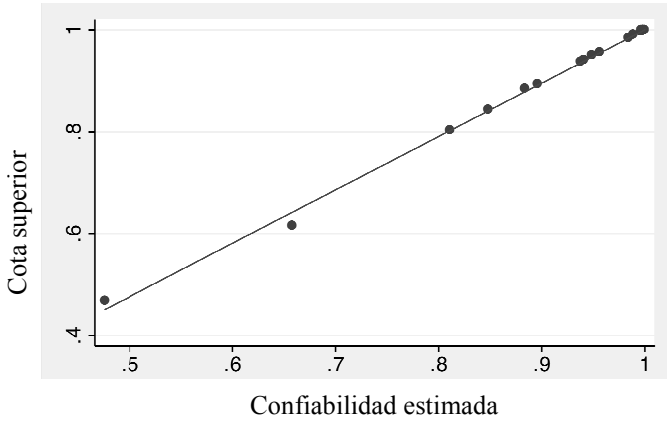

Figura 12. Diagrama de dispersión de confiabilidad estimada contra cota superior de confiabilidad

El correspondiente análisis de regresión indica una alta correlación entre la cota superior de confiabilidad y la confiabilidad estimada de la red (Tabla 3).

Tabla 3. Análisis de regresión de la cota superior de confiabilidad contra la confiabilidad estimada

Source	SS	df	MS	Number of obs =	22
Model	.392635243	1	.392635243	F(1, 20) =	7695.02
Residual	.001020492	20	.000051025	Prob > F =	0.0000
				R-squared =	0.9974
Total	.393655735	21	.018745511	Adj R-squared =	0.9973
				Root MSE =	.00714

cota_super~r	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
eval_conf	1.047261	.0119385	87.72	0.000	1.022358	1.072165
_cons	-.047233	.0111301	-4.24	0.000	-.07045	-.024016

En vista de la buena aproximación de la cota superior, en dos de los algoritmos propuestos en el siguiente capítulo esta cota se utiliza para hacer una preselección de las soluciones candidatas, de modo que solamente aquellas soluciones con las cotas más altas son enviadas a la rutina para una evaluación más precisa de su confiabilidad.

3.4 Técnicas de reducción

Estas técnicas buscan reducir el tamaño de la red original mediante la sustitución de conjuntos de arcos en serie o paralelo, preservando la confiabilidad de la red. En muchos casos el esfuerzo computacional requerido por las técnicas de simplificación se compensa con significativos ahorros de tiempo al evaluar la confiabilidad de la red resultante (Rubino, 1998).

De acuerdo con Konak (2007), en redes de confiabilidad global existen tres tipos básicos de reducciones: reducción de arcos en paralelo, reducciones de grado 1 y reducciones de grado 2.

3.4.1 Reducción de arcos en paralelo

Los arcos en paralelo entre dos nodos pueden reemplazarse por un arco de confiabilidad equivalente. Dado que en esta investigación se considera que hay dos tipos de arcos en cada conexión (los arcos preexistentes y los nuevos arcos), la confiabilidad del arco equivalente en cada conexión se determina mediante la Ecuación (1.1).

3.4.2 Reducciones de grado 1

Si un nodo i con un solo arco incidente (i,j), tanto el nodo como el arco incidente pueden ser removidos de la red original G, para obtener una red reducida G'. En este caso, $R_s(G) = \lambda R_s(G')\lambda$ (Konak, 2007), donde $\lambda = p_{(i,j)}$ es el factor de preservación de confiabilidad y $p_{(i,j)}$ es la confiabilidad del arco removido.

3.4.3 Reducciones de grado 2

Esta reducción se puede aplicar a nodos que tienen solamente dos arcos incidentes. En la red G de la Figura 13, el nodo i y sus respectivos arcos incidentes (i,j) e (i,k), pueden ser reemplazados por el arco (j,k).

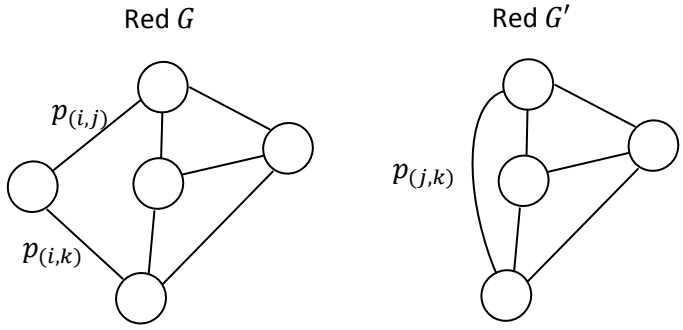

Figura 13. Reducción de grado 2

La confiabilidad del nuevo arco es (Konak, 2007),

$$p_{(j,k)} = \frac{p_{(i,j)}p_{(i,k)}}{p_{(i,j)} + p_{(i,k)} - p_{(i,j)}p_{(i,k)}} \tag{3.10}$$

La confiabilidad de la red original G se estima a partir de la confiabilidad de la nueva red G', aplicando el factor multiplicativo (Konak, 2007)

$$\lambda = \left(p_{(i,j)} + p_{(i,k)} - p_{(i,j)}p_{(i,k)}\right) \tag{3.11}$$

Y luego calculando,

$$R_s(G) = \lambda R_s\left(G'\right)$$

En general, al realizar múltiples reducciones, la confiabilidad de la red original $\hat{R}(G)$ se calcula al multiplicar el factor multiplicativo λ por la confiabilidad de la red reducida $\hat{R}(G')$. El factor λ garantiza que a pesar de las reducciones, la confiabilidad global del sistema no sea alterada.

3.4.4 Algoritmo de reducción

A partir de lo anteriormente discutido, y considerando el algoritmo de reducción de Konak (2007), se propone el siguiente algoritmo.

```
Inicio
λ = 1
Para ∀ conexion h
    p_h = 1 − (1 − r_0h)^q_h (1 − r_Ah)^x_h %% Reducción de arcos en paralelo
Fin
```

```
Mientras ∃ nodo i | g(i) = 1 ∨ g(i) = 2
    Para ∀ nodo i | g(i) = 1 %% Nodos con un arco incidente (i,j)
        Remover nodoi, arco(i,j) ∃ j
        λ = λp(i,j)
    Fin
    Para ∀ nodo i | g(i) = 2 %% Con dos arcos incidentes (i,j),(i,k)
        Remover nodoi, arco(i,j), arco(i,k) ∃ j,k
        z = p(i,j) + p(i,k) − p(i,j)p(i,k)
        p(j,k) = 1 − (1 − p(j,k))(1 − p(i,j)p(i,k)z^−1)
        λ = λz
    Fin
Fin
Fin
```

El factor de reducción de la varianza del estimador $\hat{R}(G)$, debido a las reducciones de la red, está dado por (Konak, 2007),

$$\delta = \frac{Varianzaantesdereducciones}{Varianzadespuésdereducciones} = \frac{1 - R(G)}{\lambda - R(G)} \tag{3.12}$$

Experimentalmente Konak (2007) encontró que al combinar técnicas de reducción de varianza con métodos de reducción de redes, el factor δ de reducción de varianza puede ser aún más significativo. Por lo tanto, la inclusión del algoritmo de reducción en el procedimiento de evaluación de confiabilidad de redes, no sólo permite reducir el tamaño del problema, sino que también disminuye la varianza de las estimaciones. En el Apéndice 8 se incluye el código en Matlab del algoritmo mostrado en esta sección.

3.5 Algoritmo para la evaluación de confiabilidad de redes generales no dirigidas

Ninguno de los métodos discutidos en este capítulo es adecuado para todos los tipos de red. Dado que los métodos de cálculo exacto tienden a aumentar sustancialmente los tiempos de ejecución en la medida en que aumenta el tamaño y la densidad de la red, se determinaron los siguientes criterios para determinar el método a utilizar en cada caso.

- Se utiliza enumeración completa (Sección 3.1.1) cuando el número de conexiones es menor o igual que 15, dado que el total de combinaciones de estados de las conexiones de una red con este número de nodos (32,768 combinaciones), es inferior al número de iteraciones que requeriría la Simulación Monte Carlo con una confianza del 95% y un error de 1e-3.

- Se utiliza el principio de inclusión-exclusión (Sección 3.1.3) en redes con menos de 11 árboles de expansión, lo que corresponde a 1024 posibles combinaciones de los estados de la red.
- En las redes que no cumplen ninguna de estas dos condiciones la confiabilidad se evalúa mediante simulación Monte Carlo con variables antitéticas (Sección 3.2.2). La Figura 14 muestra el algoritmo propuesto para la estimación de confiabilidad de redes generales. La implementación de este algoritmo se incluye en el Apéndice 9.

El diagrama de flujo del algoritmo se muestra en la Figura 14. Para este algoritmo se han definido las siguientes funciones:

- $C(G)$: Evaluación de conectividad de la red G
- $T(G)$: Cálculo del número de árboles de expansión de la red G
- $L(G)$: Listado de los árboles de expansión de la red G
- $R(G)$: Evaluación de la confiabilidad de la red G

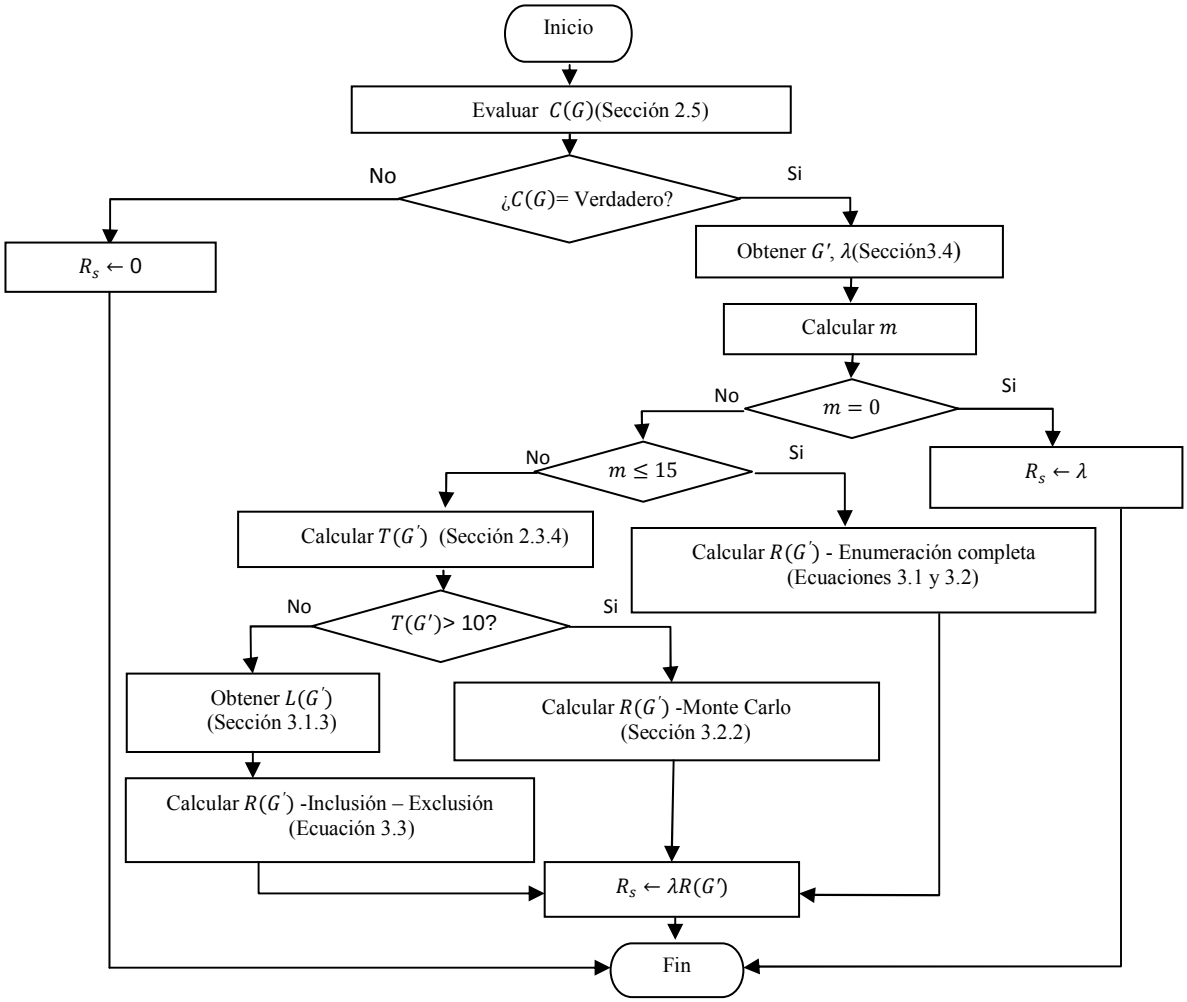

Figura 14. Algoritmo para evaluar la confiabilidad global de redes generales no dirigidas

3.6 Análisis de los tiempos de ejecución del algoritmo y de la desviación estándar de la estimación de la confiabilidad

Se realizó un experimento con variaciones controladas en dos factores: tamaño de red y número de iteraciones del algoritmo Monte Carlo. Para el número de iteraciones se definieron 5 niveles, con un tamaño mínimo de 20×10^3, y con incrementos en potencias de 4. Los niveles de los factores se detallan en la Tabla 4.

Tabla 4. Niveles de los factores considerados en el experimento

Factores	Niveles
Número de nodos	10, 15, 20, 25, 30
Número de iteraciones	20×10^3, 80×10^3, 320×10^3, 1.28×10^6, 5.12×10^6

En cada condición experimental se realizaron 2 réplicas y se registraron las variables confiabilidad calculada, tiempo de ejecución y varianza estimada (de acuerdo con la Ecuación 3.8). En cada réplica, a partir de la varianza calculada se estimó la desviación estándar correspondiente.

3.6.1 Análisis de los tiempos para la estimación de la confiabilidad

Al definir como variable de respuesta el logaritmo natural del tiempo de ejecución (medido en segundos) del algoritmo Monte Carlo, y como factores el logaritmo natural del tamaño de muestra, el número de nodos de la red y la desviación estándar de la estimación de la confiabilidad, con sus respectivas interacciones, el análisis de varianza reflejó que el logaritmo natural del número de iteraciones es el único factor significativo. Al eliminar los demás factores, se obtuvieron los resultados mostrados en la Tabla 5 (ver página 40).

De esta manera, la relación entre el número de iteraciones y el tiempo requerido por el algoritmo para estimación de la confiabilidad, puede describirse mediante la Ecuación 3.13.

$$tiempo = 4.531783 \times 10^{-5} \cdot iteraciones \tag{3.13}$$

Por consiguiente, el tiempo requerido por el algoritmo para estimación de la confiabilidad incrementa linealmente con el número de iteraciones. Para 150,000 iteraciones el algoritmo Monte Carlo demoraría aproximadamente 70 segundos para estimar la confiabilidad. Este tiempo es sustancialmente alto, si se considera que en los algoritmos de optimización que se comparan más adelante se hacen múltiples llamados a la rutina para el cálculo de la confiabilidad. Para evitar una demora excesiva en la obtención de los resultados de los algoritmos de optimización, en el capítulo 5 se utiliza una estrategia similar a la empleada por Hernández (2007), la cual consiste en incrementar gradualmente el tamaño de muestra en cada iteración del algoritmo.

Tabla 5. Análisis de varianza y análisis de regresión del tiempo de ejecución del algoritmo para el cálculo de la confiabilidad

```
                    Number of obs =      50     R-squared     =  0.9896
                    Root MSE      = .207101     Adj R-squared =  0.9893

------------------------------------------------------------------------------
      Source |  Partial SS    df       MS           F     Prob > F
-------------+----------------------------------------------------------------
       Model |  194.982324     1  194.982324    4546.04     0.0000
ln_iteraciones |  194.982324     1  194.982324    4546.04     0.0000
    Residual |  2.05875008    48  .042890627
-------------+----------------------------------------------------------------
       Total |  197.041074    49   4.0212464
------------------------------------------------------------------------------
```

```
      Source |       SS       df       MS              Number of obs =      50
-------------+------------------------------           F(  1,    48) = 4546.04
       Model |  194.982324     1  194.982324           Prob > F      =  0.0000
    Residual |  2.05875008    48  .042890627           R-squared     =  0.9896
-------------+------------------------------           Adj R-squared =  0.9893
       Total |  197.041074    49   4.0212464           Root MSE      =   .2071

------------------------------------------------------------------------------
   ln_tiempo |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
ln_iteraciones |   1.007261   .0149391    67.42   0.000     .9772242    1.037298
       _cons |  -10.00181   .1916213   -52.20   0.000    -10.3871   -9.616534
------------------------------------------------------------------------------
```

3.6.2 Análisis de la desviación estándar de la estimación de la confiabilidad

Para este análisis se consideraron los factores confiabilidad, logaritmo natural del tamaño de muestra y número de nodos de la red, con sus respectivas interacciones. El análisis de varianza indicó que los únicos factores significativos son la confiabilidad y la interacción entre la confiabilidad y el logaritmo natural del tamaño de muestra. El análisis de varianza y el análisis de regresión con estos factores se muestra en las Tablas 6 y 7.

Tabla 6. Análisis de varianza de la desviación estándar de la confiabilidad

```
                    Number of obs =      50     R-squared     =  0.7558
                    Root MSE      = .000023     Adj R-squared =  0.7399

------------------------------------------------------------------------------
      Source |  Partial SS    df       MS           F     Prob > F
-------------+----------------------------------------------------------------
       Model |  7.2693e-08     3  2.4231e-08      47.45     0.0000
             |
confiabilidad |  1.5098e-08     1  1.5098e-08      29.57     0.0000
ln_iteraciones |  9.2511e-10     1  9.2511e-10       1.81     0.1849
confiabilidad#ln_iteraciones |  1.2144e-08     1  1.2144e-08      23.78     0.0000
             |
    Residual |  2.3488e-08    46  5.1062e-10
-------------+----------------------------------------------------------------
       Total |  9.6182e-08    49  1.9629e-09
------------------------------------------------------------------------------
```

Tabla 7. Análisis de regresión de la desviación estándar de la confiabilidad

Source	SS	df	MS	Number of obs =	50
				F(2, 47) =	165.06
Model	.00054402	2	.00027201	Prob > F =	0.0000
Residual	.000077454	47	1.6480e-06	R-squared =	0.8754
				Adj R-squared =	0.8701
Total	.000621474	49	.000012683	Root MSE =	.00128

desvest	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
confiabilidad	.0356022	.0020159	17.66	0.000	.0315467	.0396577
c.ln_iteraciones# c.confiabilidad	-.0024121	.0001344	-17.94	0.000	-.0026825	-.0021416
_cons	.0007921	.0005275	1.50	0.140	-.0002691	.0018534

De esta manera, la relación entre la desviación estándar de la estimación de la confiabilidad, el número de iteraciones del algoritmo Monte Carlo y la confiabilidad estimada se puede representar mediante la ecuación,

$$\sigma = 0.0356022 \cdot confiabilidad - 0.0024121 \cdot \ln(iteraciones) \cdot confiabilidad + 0.0007921 \quad (3.14)$$

La Ecuación 3.14 es consistente con el hecho de que en redes de alta confiabilidad se requiere un gran tamaño de muestra para alcanzar estimaciones precisas (El Khadiri, et al., 1996).

4 Descripción de los algoritmos para maximizar la confiabilidad mediante la asignación de arcos

En este capítulo se describen dos métodos propuestos para resolver de forma aproximada el problema de optimización definido en la Sección 1.3, a saber

$$\text{Maximizar} \quad R_s(x_1, x_2, \dots, x_i, \dots, x_m)$$

Sujeto a:

$$\sum_{i=1}^{m} c_i x_i \leq b$$

$$x\_min_i \leq x_i \leq x\_max_i \qquad \forall\, i$$

$$x_i \in \mathbb{Z} \quad \forall\, i$$

$$x_i \geq 0$$

Donde,

m : Total de posibles conexiones de la red

x_i : Número de nuevos arcos que deben asignarse a la conexión i

c_i : Costo de cada nuevo arco asignado a la conexión i

b : Presupuesto total disponible para los nuevos arcos

x_min_i : Número mínimo de nuevos arcos en la i-ésima conexión

x_max_i : Número máximo de nuevos arcos en la i-ésima conexión

$R_s(x_1, x_2, \dots, x_i, \dots, x_m)$: Función de confiabilidad del sistema

Los métodos que se describen son un algoritmo genético y un algoritmo basado en programación lineal entera secuencial, propuesto por el Dr. Noel Artiles y desarrollado por Hernández (2007). En estos métodos se evalúa la confiabilidad mediante el algoritmo presentado en la Figura 14**,** o mediante la cota superior de confiabilidad mostrada en la Sección 3.3.

4.1 Algoritmo basado en programación lineal entera secuencial (PLES)

4.1.1 Descripción del algoritmo

Se utiliza un enfoque de superficie de respuesta para encontrar la asignación de arcos que maximiza la confiabilidad de la red. De manera iterativa, el método busca la solución del problema de optimización mediante una aproximación lineal de la función de confiabilidad de la red.

En cada iteración se ajusta un modelo de primer orden con la confiabilidad de la red como variable de respuesta y con el número de arcos en cada conexión como variables de entrada. Los parámetros del

modelo de regresión son estimados tomando como puntos experimentales: i) la red de la solución actual, ii) las posibles redes que resultan al agregar un arco en cada conexión a la red actual (sin exceder el número máximo de arcos permitidos en cada conexión), iii) las posibles redes que resultan al restar un arco en cada conexión a la red actual (sin incumplir con el número mínimo de arcos en cada conexión), y vi) la red con el máximo número de arcos en cada conexión.

El modelo de regresión obtenido es utilizado como función objetivo en un modelo de optimización entero que tiene como restricciones i) el presupuesto disponible, ii) el número mínimo y máximo de arcos en cada conexión.

Para reducir el error por la aproximación lineal, en este modelo se permite únicamente la adición o sustracción de un arco en cada conexión. La solución del modelo de optimización constituye la red de la solución actual. Este proceso se repite hasta lograr convergencia (Hernández, 2007). Para este método se establecen dos criterios de parada: número máximo de iteraciones (max_iter) y número de iteraciones en que se ha obtenido la misma solución (max_rep_sol).

El algoritmo correspondiente a este método es:

1. Defina el vector $\mathbf{w}_{(\mathbf{1}\times\boldsymbol{m})} = \mathbf{x\_min}_{(\mathbf{1}\times\boldsymbol{m})}$
2. Construya la matriz $\mathbf{x}_* = \begin{bmatrix} \mathbf{w}_{(\mathbf{1}\times\boldsymbol{m})} \\ \vdots \\ \mathbf{w}_{(\mathbf{1}\times\boldsymbol{m})} \end{bmatrix}_{(\boldsymbol{m}\times\boldsymbol{m})}$
3. $\mathbf{x}^+ = \mathbf{x}_{*_{(\boldsymbol{m}\times\boldsymbol{m})}} + \mathbf{I}_{(\boldsymbol{m}\times\boldsymbol{m})}$
4. En la matriz $\mathbf{x}_+$ suprima las filas donde se excede el número máximo de arcos en al menos una de las conexiones.
5. $\mathbf{x}^- = \mathbf{x}_{*_{(\boldsymbol{m}\times\boldsymbol{m})}} - \mathbf{I}_{(\boldsymbol{m}\times\boldsymbol{m})}$
6. En la matriz $\mathbf{x}^+$ suprima las filas donde no se cumple con el número mínimo de arcos en al menos una de las conexiones.
7. $\mathbf{X} = \begin{bmatrix} \mathbf{1} & \mathbf{x_0} \\ \vdots & \mathbf{x}^+ \\ \vdots & \mathbf{x}^- \\ \mathbf{1} & \mathbf{x\_max} \end{bmatrix}$
8. Eliminar las filas repetidas de la matriz
9. Calcular **Y**, donde el *i*-ésimo elemento de **Y** es la confiabilidad $\boldsymbol{R_s}$ correspondiente a la *i*-ésima fila de la matriz **X**. Para este heurístico la confiabilidad se calcula mediante el algoritmo general para la evaluación de la confiabilidad global (Sección 3.5).
10. Calcular los parámetros del modelo de regresión,

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Y}$$

11. Resolver el problema de optimización:

$$Max\hat{R}_s(\mathrm{x}) = \hat{\beta}_0 + \sum_{i=1}^{m} \hat{\beta}_i x_i \tag{4.1}$$

Sujeto a:

$$\sum_{i=1}^{m} c_i x_i \leq b \tag{4.2}$$

$$\max(x\_min_i, w_i - 1) \leq x_i \leq \min(x\_max_i, w_i + 1) \tag{4.3}$$

$$x_i \in \mathbb{Z}$$

$$x_i \geq 0$$

12. Evaluar los criterios de terminación. Si se cumplen, termina del proceso. De lo contario, se asigna $\mathbf{w}_{(\mathbf{1}\times \boldsymbol{m})} \leftarrow \mathbf{x}$ y se regresa al paso 2.

La implementación de este método en Matlab se incluye en el Apéndice 10.

4.1.2 Ejemplo numérico del algoritmo PLES

Considérese el problema de asignar arcos a las conexiones de una red con 2 arcos entre los nodos 3 y 4, y un arco entre los nodos 2 y 4, con un presupuesto de 50. La configuración actual de esta red se muestra en la Figura 15.

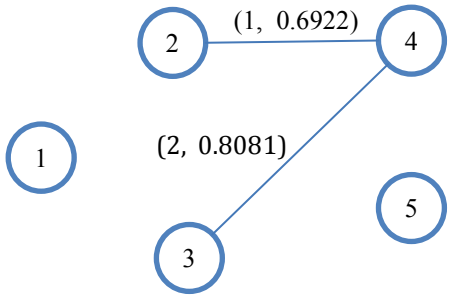

Figura 15. Arcos de la red original (pre-existentes) para el ejemplo

El número máximo (x_{max}) y el número mínimo (x_{min}) de arcos en cada conexión, y la confiabilidad ($\mathrm{r_A}$) y los costos (c) de los nuevos arcos se muestran en la Tabla 8.

Tabla 8. Datos de la red para el ejemplo

Conexiones	1-2	1-3	2-3	1-4	2-4	3-4	1-5	2-5	3-5	4-5
Máximo	3	0	3	4	0	1	0	1	3	2
Mínimo	1	0	2	1	0	0	0	0	1	1
Confiabilidad	0.5549	0.7725	0.8879	0.647	0.6888	0.8382	0.7155	0.9489	0.6315	0.7026
Costo	3	0	3	4	0	1	0	1	3	2

Al ejecutar el algoritmo con max_iter = 50 y max_rep_sol = 5, se obtuvieron los resultados de la Tabla 9 y la Figura 16.

Tabla 9. Resultados del algoritmo basado en programación lineal entera secuencial, con los datos de la Tabla 8

Iteracion	Tiempo Acumulado (segundos)	Confiabilidad	Costo	Solución									
				1-2	1-3	2-3	1-4	2-4	3-4	1-5	2-5	3-5	4-5
1	0.28	0.97463	33	3	0	2	3	0	0	0	1	1	1
2	0.38	0.99046	42	3	0	2	4	0	0	0	1	2	2
3	0.6	0.99840	48	3	0	3	4	0	0	0	1	3	2
4	0.73	0.99840	48	3	0	3	4	0	0	0	1	3	2
5	0.84	0.99840	48	3	0	3	4	0	0	0	1	3	2
6	0.95	0.99840	48	3	0	3	4	0	0	0	1	3	2
7	1.07	0.99840	48	3	0	3	4	0	0	0	1	3	2
8	1.19	0.99840	48	3	0	3	4	0	0	0	1	3	2
9	1.31	0.99840	48	3	0	3	4	0	0	0	1	3	2
10	1.43	0.99840	48	3	0	3	4	0	0	0	1	3	2

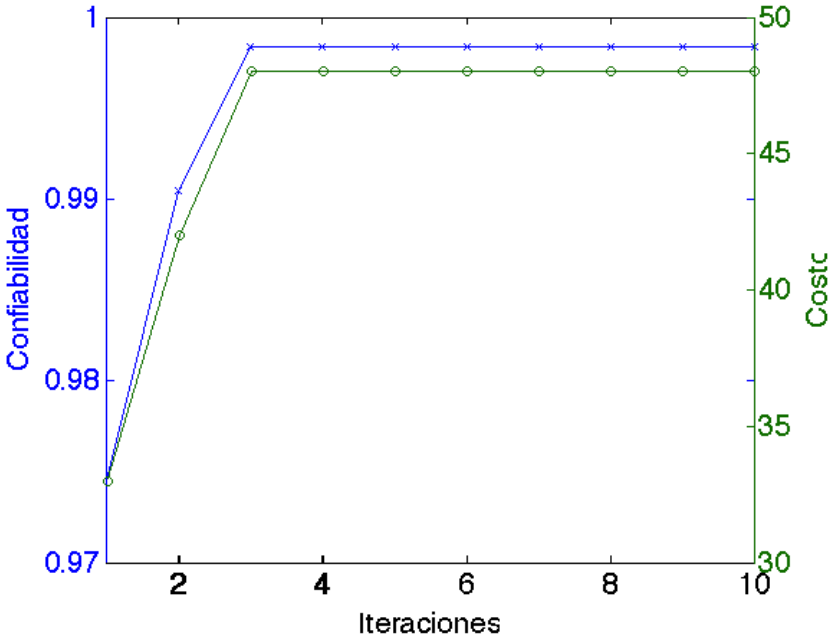

Figura 16. Costos y confiabilidad de la red a lo largo de las iteraciones en el ejemplo

Como se observa en la Figura 16, a partir de la tercera iteración se estabilizó el resultado del algoritmo PLES.

4.2 Algoritmo genético

Desde principios de los 90's los algoritmos genéticos empezaron a ser adoptados para la solución de problemas de confiabilidad. Actualmente son una de las técnicas con más auge en la solución de problemas de confiabilidad de redes, gracias a su desempeño en problemas combinatorios y en general, cuando la función objetivo es no derivable o es difícil de obtener analíticamente, como típicamente ocurre en redes complejas. Algunas ventajas de los algoritmos genéticos son (Haupt & Haupt, 2004, p. 23):

- Trabajan con un gran número de variables
- Admiten variables discretas
- Exploran simultáneamente una muestra amplia del espacio de solución
- Son adecuados para computación paralela
- Funcionan aún cuando la función objetivo es compleja
- Trabajan con datos experimentales o funciones analíticas

Los algoritmos genéticos hacen parte del campo de la computación evolutiva (Melanie, 1996). Su proceso imita los fenómenos de evolución y selección natural. Por ello, contienen poblaciones de cromosomas, realizan selección de acuerdo con la aptitud, generan descendencia mediante cruces genéticos, y realizan mutación aleatoria de los nuevos descendientes.

El procedimiento general del algoritmo genético, es el siguiente. Aleatoriamente o mediante un método heurístico se genera una población inicial de cromosomas. A partir dicha población se obtiene una nueva población, realizando los siguientes pasos (Melanie, 1996):

1) Se calcula la función de aptitud de cada elemento de la población.
2) A partir de la población actual se selecciona aleatoriamente con sustitución un par de cromosomas padre, de manera que los elementos más aptos tengan mayor probabilidad de ser seleccionados.
3) Los cromosomas seleccionados se cruzan o no, de acuerdo con la probabilidad de cruce genético p_c ("*crossover"*).
4) Los dos descendientes mutan con una probabilidad p_m. Si el total de elementos obtenidos es impar, se descarta uno de los cromosomas de manera aleatoria.
5) Se regresa al paso (3) hasta que se haya obtenido la descendencia que hará parte de la nueva población
6) Se reemplaza la anterior población con la nueva población obtenida
7) Si no se cumplen los criterios de parada, se regresa al paso (1). Los posibles criterios de parada son: se alcanza el máximo número de generaciones, la solución no mejora después de varias iteraciones, se ha excedido el tiempo de ejecución, entre otros.

Cada uno de los cromosomas de la población final, constituye una posible solución. Típicamente se escoge entre ellas la solución con mejor aptitud. Sin embargo, al contar con un conjunto final de posibles soluciones, se puede evaluar cuál de ellas desde el punto de vista práctico es más conveniente de implementar. De esta manera, se puede contar con varias alternativas de configuración para la red Kuo *et al.* (2001, p. 63).

4.2.1 Consideraciones sobre el diseño del algoritmo genético

Para lograr un desempeño efectivo del algoritmo genético, la codificación de las soluciones, el método de evaluación de aptitud de las soluciones y los operadores genéticos deben ser definidos de tal manera que se asegure una adecuada representación del problema y que se logre un justo balance entre la exploración y la explotación del espacio de soluciones.

Para la codificación de las soluciones se utilizó un vector de enteros en el que cada posición representa el número de nuevos arcos asignados a cada posible conexión de la red. De esta manera, el vector tendrá tantas posiciones como conexiones posibles tenga la red. En general, para una red con $\boldsymbol{n}$ nodos, el vector solución tendrá $n(n-1)/2$ posiciones.

Para la función de aptitud se dispone de un algoritmo descrito en la Sección 3.5. De acuerdo con las características de la red, este algoritmo evalúa condiciones necesarias y suficientes de conectividad, reduce la red y selecciona el método más indicado para la evaluación de su confiabilidad. También, como se discutió en la sección Sección 3.3, la cota superior puede ofrecer una buena aproximación de la confiabilidad de la red. Por lo tanto, para la función de aptitud se pueden considerar dos opciones: mediante el algoritmo general de evaluación de confiabilidad (Sección 3.5) o mediante la cota superior para evaluación de confiabilidad (Sección 3.3) y aplicando el algoritmo de evaluación de confiabilidad (Sección 3.5) solamente en las soluciones cuya cota superior supera el mejor valor de la solución actual. De esta manera se podrían evitar los llamados a rutinas para la evaluación de confiabilidad, en soluciones que no lograrán superar al elemento con mayor aptitud. El tiempo de ejecución de algoritmos genéticos con estas dos variantes de la función de aptitud se analiza en el siguiente capítulo.

Para el problema de optimización de confiabilidad de redes, la utilización de operadores genéticos de tipo general podría conducir a la obtención de soluciones ilegales o infactibles, lo cual no solamente disminuiría la calidad de la población, sino que podría incrementar sustancialmente los tiempos de ejecución del algoritmo genético. Por consiguiente, para el problema analizado se desarrollaron operadores genéticos especializados, tanto para el cruce genético y la mutación, como para generar la población inicial de soluciones. Estos operadores aseguran que en todas las soluciones que de ellos se derivan, se cumplan las siguientes condiciones: 1) los valores de todas las posiciones del vector son enteros, 2) la cantidad de arcos en cada conexión está dentro del número mínimo y máximo de arcos

permitidos en la conexión respectiva, 3) el costo total no excede el presupuesto disponible para la solución, 4) la solución representa una red conectada.

4.2.2 Descripción del algoritmo implementado

A continuación se describen los operadores genéticos desarrollados y el procedimiento general del algoritmo implementado.

- **Población inicial**

Se genera aleatoriamente una población en la que cada solución cumple con las restricciones de número máximo y número mínimo de nodos en cada conexión, y presupuesto disponible. Adicionalmente, el algoritmo busca eliminar soluciones repetidas y soluciones con una baja cota de confiabilidad superior. El procedimiento es el siguiente:

1. Se genera una población de dos veces el tamaño de la población requerida.
2. Se suprimen los elementos repetidos en la población obtenida.
3. Si el tamaño de la población resultante es menor que el tamaño de la población requerida, se llama a la rutina para generación de población para obtener los elementos restantes.
4. Si, por el contrario, el tamaño de la población resultante es mayor que la población requerida, para cada elemento de la población se calcula la cota superior de confiabilidad (ver Sección 3.3). A continuación se eliminan los elementos con la cota superior más baja.

En los pasos 1 y 3 se utiliza la siguiente subrutina para generar aleatoriamente los elementos de una población de tamaño N:

```
j ← 1
while j ≤ N
    for i=1 hasta m
          r ← aleatorio()
          e_i ← ⌊r * (x_max_i − x_min_i)⌋ + x_min_i
    end
    if conectividad([e_1, e_2, …, e_m])=true and [e_1, e_2, …, e_m] × [c_1, c_2, …, c_m]′ ≤ b
          poblacion(j,:) ←[e_1, e_2, …, e_m];
          j=j+1;
    end
end
```

- **Selección**

La selección de los padres para la nueva generación se realiza mediante el método de torneo. Aleatoriamente se escoge un número de individuos, de los cuales se selecciona como padre el elemento con mayor aptitud.

- **Mutación**

La mutación se realiza mediante una función aleatoria gaussiana, la cual utiliza una variable aleatoria normal para asignar a cada posición del vector seleccionado un valor entero entre el mínimo y el máximo número de arcos en cada conexión. Mediante un factor k de escala de mutación, se reduce la dispersión en la medida en que transcurren las generaciones,

$$k = \left(e - \frac{t \cdot Ng}{Nt}\right)(x\_max_i - x\_min_i) \tag{4.4}$$

Donde,

e: Factor inicial de escala

t : Factor de contracción

Ng: Número de la generación actual, $Ng = 1, \dots Nt$

Nt: Número máximo de generaciones

El valor con la mutación x_i' en cada elemento del vector se obtiene al sumar la parte entera del producto entre el factor de escala k y un número aleatorio normal z, es decir,

$$x_i^{'} = x_i + \text{Round}[k \cdot z] \tag{4.5}$$

Adicionalmente, se debe corregir el valor de x_i', en caso de que no se cumpla con el límite del número de arcos en la conexión. Por lo tanto,

$$x_i^{'} = \begin{cases} x\_max_i, & \text{si} \quad x_i^{'} > x\_max_i \\ x\_min_i, & \text{si} \quad x_i^{'} > x\_min_i \end{cases}$$

- **Reproducción**

En cada generación se obtiene la *élite*, los elementos de la población con mayor aptitud. Estos elementos se trasladan a la siguiente generación. El resto de la población de la siguiente generación se obtiene mediante cruce genético y mutación. La fracción de mutación corresponde a la fracción de la nueva generación que se obtendrá mediante cruce genético.

- **Cruce**

Los padres son seleccionados por torneo. Inicialmente se utiliza una función de cruce disperso, la cual utiliza un vector aleatorio binario para seleccionar los elementos de ambos padres. Si la red que se obtiene a partir del cruce no está conectada o excede el presupuesto, se descarta el elemento obtenido y se realiza un cruce de punto único. Para este cruce, se define aleatoriamente un punto de corte. Del primer padre se toma la primera parte del vector hasta la posición seleccionada, y del segundo padre se toma el segmento restante del vector. De nuevo, si no se obtiene una solución factible, se descarta la solución y se realiza un cruce aritmético, el cual asigna al nuevo elemento el valor redondeado de la media aritmética entre los elementos de los dos padres. Si en este punto del algoritmo aún es infactible, se seleccionan dos nuevos padres y se vuelve a realizar el cruce. El algoritmo se repite hasta que se ha obtenido toda la descendencia para la nueva generación.

- **Criterios de parada**

Se utilizan dos criterios de parada: el número máximo de generaciones y el número de generaciones estancadas. Este último criterio corresponde al número de generaciones en que el cambio en la solución está por debajo de una tolerancia.

El código en Matlab del algoritmo genético descrito se incluye en el Apéndice 11.

4.2.3 Ejemplo numérico del algoritmo genético para optimización de confiabilidad de redes

Con los mismos datos del ejemplo de la Sección 4.1.2 (Figura 15 y Tabla 8) se probó el algoritmo genético con los parámetros dados en la Tabla 10.

Tabla 10. Parámetros del algoritmo genético para la red del ejemplo

Parámetro	Valor
Tamaño de población	20
Máximo número de generaciones	20
Máximo número de generaciones con estancamiento	10
Fracción de cruce	0.7
Tamaño del torneo	4
Tamaño de élite	2
Tolerancia	1.00E-05
Escala de cruce genético	0.4
Factor de contracción	1

Los resultados se muestran en la Figura 17 y la Tabla 11.

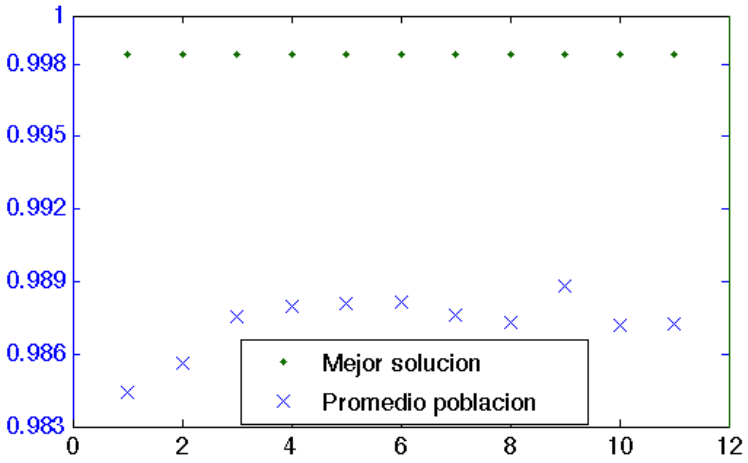

Figura 17. Resultados del algoritmo genético para optimización de confiabilidad de redes

Tabla 11. Resultados del algoritmo genético a lo largo de 11 generaciones

Iter.	Tiemp. (segundos)	Conf. Prom	Conf. Mejor	Costo	Solución									
					1-2	1-3	2-3	1-4	2-4	3-4	1-5	2-5	3-5	4-5
1	0.12	0.98442	0.99838	46	3	0	2	4	0	1	0	1	3	2
2	0.16	0.98561	0.99838	46	3	0	2	4	0	1	0	1	3	2
3	0.2	0.98756	0.99838	46	3	0	2	4	0	1	0	1	3	2
4	0.23	0.98796	0.99840	49	3	0	3	4	0	1	0	1	3	2
5	0.27	0.98807	0.99840	49	3	0	3	4	0	1	0	1	3	2
6	0.3	0.98813	0.99838	46	3	0	2	4	0	1	0	1	3	2
7	0.34	0.98762	0.99838	46	3	0	2	4	0	1	0	1	3	2
8	0.37	0.98733	0.99840	49	3	0	3	4	0	1	0	1	3	2
9	0.41	0.98884	0.99840	49	3	0	3	4	0	1	0	1	3	2
10	0.45	0.98721	0.99840	49	3	0	3	4	0	1	0	1	3	2
11	0.48	0.98728	0.99840	49	3	0	3	4	0	1	0	1	3	2

Gracias al heurístico para la generación de la población inicial, la solución óptima ya estaba contenida en la primera generación. El resultado obtenido coincide con la solución encontrada mediante el algoritmo de programación lineal entera secuencial. El corto tiempo que transcurre para completar las 11 generaciones del algoritmo se explica en parte por el uso del algoritmo presentado en la Sección 2.5.

4.3 Implementación de los algoritmos

Todos los algoritmos fueron codificados en Matlab. Para reducir el tiempo requerido para la evaluación de la confiabilidad, se hizo una implementación paralelizada del algoritmo Monte Carlo

mediante el *toolbox* de computación paralela. En un computador con procesador Intel i5 de doble núcleo, los tiempos de ejecución de los algoritmos paralelizados fueron aproximadamente un 50% inferiores a los tiempos de los algoritmos sin la paralelización.

Para el algoritmo de programación lineal entera secuencial (PLES) se utilizaron la función de regresión y el *toolbox* de optimización para resolver el modelo lineal. Los algoritmos genéticos fueron programados sin utilizar el *toolbox* de optimización ni el *toolbox* de algoritmos genéticos y búsqueda directa.

En el capítulo siguiente se describen y analizan los experimentos computacionales realizados con estos dos métodos, a fin de comparar su desempeño en diferentes condiciones (tamaño de red, densidad de la red, etc.).

5 Experimentos computacionales

En este capítulo se evalúa el desempeño de tres heurísticos para la asignación óptima de arcos:

- PLES: Algoritmo basado en Programación Lineal Entera Secuencial (Sección 4.1). Para este heurístico la confiabilidad se calcula mediante el algoritmo general para la evaluación de la confiabilidad global (Sección 3.5).
- AG1: Algoritmo Genético (Sección 4.2.2). La aptitud se evalúa mediante el algoritmo de la Sección 3.5.
- AG2: Algoritmo Genético (Sección 4.2.2), con la aptitud evaluada mediante la cota superior de confiabilidad (Sección 3.3). En los elementos de la población con una cota superior mayor que la mejor solución actual, la aptitud se evalúa mediante el algoritmo de la Sección 3.5.

Debido a que los algoritmos AG1 y AG2 realizan una búsqueda probabilística, y a que el método Monte Carlo (Sección 3.2) es utilizado en el algoritmo general para la evaluación de confiabilidad, puede haber diferencias en las soluciones encontradas por los tres métodos. En este capítulo se analiza si tales diferencias son significativas. También se estudian los tiempos de ejecución, con el fin de determinar cuál de estos métodos alcanza el mejor desempeño. Para el diseño de los experimentos se utilizó una metodología similar a la descrita por Hernández (2007).

5.1 Diseño experimental para comparación de los heurísticos de solución

En los experimentos para comparar el desempeño de los tres algoritmos de optimización se analizan dos variables de respuesta: la confiabilidad de la mejor solución y el tiempo de ejecución. En los experimentos se consideraron como factores el número de nodos de la red, la densidad de la red, la confiabilidad promedio de los arcos y el algoritmo o método utilizado en la solución. En este capítulo, lo que más interesa evaluar es el efecto del factor "método" en las dos variables de respuesta. Los demás factores son considerados porque ellos pueden afectar tanto la confiabilidad como el tiempo de solución. A cada combinación de niveles de los factores se le denomina condición experimental. Los factores y niveles utilizados en el experimento se indican en la Tabla 12. En cada condición experimental se generaron dos instancias (es decir, dos réplicas), cada una de las cuales fue sometida a los tres métodos. En el análisis de los resultados, los tres factores número de nodos de la red, densidad de la red y confiabilidad promedio de los arcos fueron tratados como variables continuas.

Tabla 12. Factores y niveles del experimento para comparar la calidad de las soluciones

Factor	Niveles
Densidad ($densidad$)	2/3, 1/3, 1/6, 1/12, 1/24, 1/48
Confiabilidad promedio arcos (p_{prom})	0.05, 0.45, 0.85
Tamaños de red (n)	15, 30, 45
Métodos ($metodo$)	PLES, AG1, AG2 (1, 2 y 3, respectivamente)

A partir de la confiabilidad promedio p_{prom} de los arcos se determina la confiabilidad mínima y máxima de los arcos en las conexiones, p_{min} y p_{max}, respectivamente. En este experimento se definió un rango de 0.1 entre p_{min} y p_{max}.

5.2 Procedimiento para la generación de instancias

En el área de optimización matemática, se conoce como **instancia** de un problema al par (*X, f*), donde *f* es la función objetivo del problema (también conocida como función de costo o función de utilidad) y donde *X* representa el dominio de la función, es decir el conjunto de puntos factibles definidos por las restricciones del problema de optimización (Papadimitriou, et al., 1998). En otras palabras, una instancia contiene la información para un caso específico del problema de optimización. Por lo tanto, dada una instancia (*X, f*), el problema de maximización de confiabilidad consiste en encontrar una solución $x \in X$ tal que $f(x) \geq c(y)$ para todo $y \in X$.

Para la evaluar el desempeño de los algoritmos descritos en el Capítulo 4 fue necesario generar instancias del problema. Para conformar cada instancia se generaron aleatoriamente los datos correspondientes a la confiabilidad de los arcos de la nueva red, el número mínimo y el número máximo de arcos en cada posición de la red, el costo unitario de los arcos en cada conexión, el presupuesto disponible para la nueva red, la confiabilidad de los arcos de la red existente, y el número de arcos en cada conexión de la red original. Debido a que los arcos de la red son no dirigidos (ver Sección 1.4), se deben generar matrices simétricas con las confiabilidades y los costos de los arcos y con el número de arcos en cada conexión de la red original.

A partir de los factores número de nodos n, densidad de la red d_n y confiabilidades mínima y máxima p_{min} y p_{max} de los arcos de la red, las instancias fueron generadas mediante el procedimiento que se describe a continuación.

1. Confiabilidad de los arcos de la nueva red: Para cada conexión de la red se genera un número aleatorio uniforme $U(p_{min}, p_{max})$, el cual determina la confiabilidad de los arcos de la red en la conexión respectiva.
2. Confiabilidad de los arcos de la red existente: Igualmente, se determina mediante un número aleatorio uniforme $U(p_{min}, p_{max})$.

3. Número de arcos de la red original: En cada conexión se genera un número aleatorio Bernoulli con probabilidad de éxito d_n. Si ocurre un fracaso, el número de arcos de la red original en la conexión respectiva es cero. De lo contrario, el número de arcos iniciales en dicha conexión es definido por un número aleatorio uniforme entero entre 1 y n_{max}, donde n_{max} representa el número máximo de arcos que se puede asignar a cualquiera de las conexiones. Se utilizó $n_{max} = 5$.
4. Número máximo de arcos en cada conexión (x_max_i): En cada conexión se genera un número aleatorio con probabilidad de éxito d_n. En las conexiones donde ocurrió un éxito, se genera un número aleatorio uniforme entero entre 1 y n_{max}. En las demás posiciones, el número máximo de arcos es igual a cero.
5. Número mínimo de arcos en cada conexión (x_min_i): En las posiciones donde el número máximo de arcos es mayor o igual que 1, el número mínimo de arcos en la conexión respectiva se determina mediante un número aleatorio uniforme entero entre cero y uno.
6. Se evalúa la conectividad de la red conformada por la unión de los arcos existentes con el máximo número de arcos nuevos. Si no existe conectividad, d_0 se incrementa en un 10% y se regresa al paso 3. De lo contrario, se continúa con el paso 7.
7. Con un número aleatorio uniforme entero entre $costo_{min}$ y $costo_{max}$ se determina el costo unitario de los nuevos arcos en cada conexión. Tomando como referencia los parámetros utilizados por Hernández (2007), se utilizó $costo_{min} = 10$ y $costo_{max} = 30$.
8. El presupuesto disponible para los nuevos arcos se determina mediante la ecuación (Hernández, 2007),

$$b = round\left(0.8\sum_{i=1}^{m} c_i(x_{max_i} - x_{min_i})\right) \tag{5.1}$$

9. Con el fin de constatar si es posible obtener una solución que cumpla con el presupuesto asignado, aleatoriamente se genera una red que cumpla las condiciones del mínimo y el número máximo de arcos en cada conexión.
10. Si la unión de la red aleatoria de nuevos arcos (recién obtenida) con la red original cumple las condiciones de factibilidad, es decir, si existe conexión y se cumple el presupuesto, los datos de la red se utilizan para la comparación de los tres algoritmos. De lo contrario, se utiliza nuevamente la Ecuación 5.1 para calcular el presupuesto disponible, y se regresa al paso 9.

Para ilustrar mejor el procedimiento anteriormente descrito, considérese el caso de generar una red de tamaño $n = 15$, con confiabilidad promedio de los arcos $p_{prom} = 0.45$ y densidad $d_n = 1/6$.

Para generar los datos correspondientes a la confiabilidad de los arcos de la nueva red, se crea una matriz de tamaño $n = 15$. Dado que la densidad promedio es $p_{prom} = 0.45$, y la diferencia entre p_{min} y p_{max} es 0.1, se sigue que $p_{min} = 0.4$ y $p_{max} = 0.5$. Por lo tanto, a cada posición de la matriz triangular superior se asigna un número aleatorio uniforme $U(0.4, 0.5)$. La matriz de confiabilidades de los arcos se obtiene al sumar la matriz obtenida con su matriz transpuesta. De igual forma se generan las confiabilidades de los arcos de la red existente.

Para generar el número de arcos de la red original, se crea una matriz de tamaño $n = 15$. Para determinar los valores de en cada posición de la matriz triangular superior, se genera un número aleatorio uniforme. Si dicho número es mayor que $d_n = 1/6$, se asigna un valor de cero a dicha posición de la matriz, con lo cual se indica que no hay arcos en la conexión respectiva de la red inicial. Si, por el contrario, el número aleatorio es menor que 0.45, la cantidad de arcos iniciales en dicha conexión es definida por un número aleatorio uniforme entero entre 1 y $n_{max} = 5$. La matriz con el total de arcos de la red original se obtiene al sumar la matriz triangular superior obtenida, con su matriz transpuesta.

Para determinar el número máximo de arcos en cada conexión, se procede de manera similar. Para cada conexión se genera un número aleatorio. En las conexiones donde dicho número es menor que d_n se genera un número aleatorio uniforme entero entre 1 y n_{max}. En las demás posiciones, el número máximo de arcos es igual a cero. Puesto que esta matriz también es simétrica, se debe sumar la matriz triangular superior obtenida, con su matriz transpuesta.

Para obtener el número mínimo de arcos en cada conexión, en las posiciones donde el número máximo de arcos es mayor o igual que 1, el número mínimo de arcos en la conexión respectiva se determina mediante un número aleatorio uniforme entero entre cero y uno. Nuevamente, debido a la simetría de la matriz, solamente es necesario generar los valores correspondientes a la matriz triangular superior, para luego sumar esta matriz con su matriz transpuesta.

Antes de generar los demás datos de la instancia, se evalúa si es posible obtener una red conectada a partir de la unión de los arcos de la red original, con los nuevos arcos que se asignen a las conexiones de la red. Para ello, en cada posición de la matriz se suma el número máximo de nuevos arcos en la posición respectiva, con el número de arcos de la red original, y se determina la conectividad de la red resultante. Si no existe conectividad, la densidad d_n se incrementa en un 10% y se vuelve a generar las matrices con el número de arcos en la red original, y con el número máximo y número mínimo de arcos en cada conexión. Si, por el contrario, existe conexión en la red resultante al unir los arcos de la red original con el número máximo de arcos en la nueva red, se procede a determinar el costo de los arcos, como se describe a continuación.

A cada posición de la matriz triangular superior se asigna un número aleatorio uniforme entero entre $costo_{min} = 10$ y $costo_{max} = 30$. La matriz de costos unitarios se obtiene al sumar esta matriz con su matriz transpuesta.

A partir de la Ecuación 5.1 se determina el presupuesto total disponible para los nuevos arcos.

Aleatoriamente se genera una red que cumple con el número de mínimo y el número máximo de arcos en cada conexión. Si esta red no cumple las dos condiciones (conectividad y cumplimiento del presupuesto), se aplica nuevamente la Ecuación 5.1 y se genera otra red de prueba. En caso contrario se considera que los datos de la instancia son adecuados para comparar los tres algoritmos.

5.3 Parámetros de los algoritmos de optimización

Para el número de iteraciones del algoritmo de evaluación de confiabilidad se utilizó una estrategia similar a la empleada por Hernández (2007), la cual consiste en incrementar gradualmente el número de iteraciones del algoritmo (Tabla 13). En los algoritmos genéticos se utilizó la siguiente fórmula para determinar el tamaño de muestra del algoritmo Monte Carlo en cada generación:

$$Ni = N_0 + \text{round}\left((N_0 - N_{max})\left(\frac{N_g}{N_t}\right)\right) \tag{5.2}$$

Donde Nt es el número máximo de generaciones, Ng es el número de la generación actual, N_0 es el tamaño de muestra de la primera generación y N_{max} es el tamaño de muestra en la última generación (de acuerdo con el número máximo de generaciones).

Se realizaron varias corridas de prueba para seleccionar los parámetros de los algoritmos. Los valores de los parámetros utilizados para la comparación de los algoritmos se muestran en la Tabla 13 (ver página 57).

5.4 Resultados de la comparación de los algoritmos

Se efectuaron las comparaciones de los tres algoritmos en un computador MacBook Pro con procesador Intel Core i5 de 2.4 GHz, con 4 GB de memoria RAM. Los datos obtenidos a partir de los experimentos computacionales se incluyen en los Apéndices 12 y 13. Los resultados se analizan a continuación.

Tabla 13. Parámetros de los heurísticos

	PLES	AG1	AG2 (Con cotas de confiabilidad)
Tamaño de población	-	n (Nodos)	$1.2n$
Máximo número de generaciones (iteraciones)	30	25	25
Máximo número de generaciones (iteraciones) con estancamiento	6	5	7
Número máximo de soluciones repetidas	4	5	5
Fracción de cruce		0.8	0.7
Tolerancia inicial	0.05	0.05	0.05
Factor de contracción de la tolerancia	-	0.5	0.5
Escala inicial de mutación	-	0.5	0.6
Factor de contracción de la mutación	-	1	1
Tamaño inicial de muestra - Monte Carlo	10,000	10,000	30,000
Tamaño final de muestra – Monte Carlo	-	120,000	150,000
Tasa de incremento del tamaño de muestra	5,000	Ecuación 5.2	Ecuación 5.2

5.4.1 Análisis de diferencias en la confiabilidad de las soluciones

Para una estimación más precisa de la confiabilidad de la solución dada por cada uno de los tres heurísticos, se calculó la confiabilidad de cada solución,. Para facilitar el cumplimiento del supuesto de distribución normal de los errores, a las confiabilidades calculadas ($conf$) se les aplicó la transformación de seno inverso (Bartlett, 1947),

$$conf\_transf = \text{arcsen}\left(\sqrt{conf}\right) \tag{5.3}$$

Inicialmente se hizo el análisis de varianza del modelo que contiene todas las interacciones,

$$conf\_transf = \beta_0 + \beta_1 densidad + \beta_2\, p_{prom} + \beta_3\, nodos + \begin{cases}\beta_4 \text{ si } metodo = 2\\ \beta_5 \text{ si } metodo = 3\end{cases} + \begin{cases}\beta_6 nodos \text{ si } metodo = 2\\ \beta_7 nodos \text{ si } metodo = 3\end{cases} \tag{5.4}$$

El análisis de varianza (Tabla 14) indicó que los factores $metodo$ y la interacción $metodo \cdot nodos$ no son significativos. Al eliminar estos factores se obtuvieron los resultados mostrados en las Tablas 15 y 16.

Tabla 14. Análisis de varianza de las confiabilidades correspondientes a la mejor solución de cada método.

```
      Number of obs =     324     R-squared     =  0.6964
      Root MSE      = .356562     Adj R-squared =  0.6897
```

Source	Partial SS	df	MS	F	Prob > F
Model	92.1512676	7	13.1644668	103.55	0.0000
densidad	39.6979826	1	39.6979826	312.25	0.0000
conf_arcos	51.2426531	1	51.2426531	403.05	0.0000
nnodos	1.19168572	1	1.19168572	9.37	0.0024
metodo	.003256325	2	.001628162	0.01	0.9873
metodo#nnodos	.000063349	2	.000031674	0.00	0.9998
Residual	40.1750513	316	.127136238		
Total	132.326319	323	.409679006		

Tabla 15. Análisis de varianza de las confiabilidades correspondientes a la mejor solución de cada método

```
      Number of obs =     324     R-squared     =  0.6963
      Root MSE      =  .35441     Adj R-squared =  0.6934
```

Source	Partial SS	df	MS	F	Prob > F
Model	92.1323214	3	30.7107738	244.50	0.0000
densidad	39.6979826	1	39.6979826	316.05	0.0000
conf_arcos	51.2426531	1	51.2426531	407.96	0.0000
nnodos	1.19168572	1	1.19168572	9.49	0.0022
Residual	40.1939976	320	.125606242		
Total	132.326319	323	.409679006		

Tabla 16. Análisis de regresión de las confiabilidades correspondientes a la mejor solución de cada método

conf_tranf	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
densidad	1.55166	.0872806	17.78	0.000	1.379943	1.723376
conf_arcos	1.217668	.0602863	20.20	0.000	1.09906	1.336276
nnodos	.0049518	.0016076	3.08	0.002	.0017889	.0081147
_cons	-.0645702	.0617594	-1.05	0.297	-.1860758	.0569355

Skewness/Kurtosis tests for Normality

Variable	Obs	Pr(Skewness)	Pr(Kurtosis)	joint adj chi2(2)	joint Prob>chi2
res_conf4	324	0.0263	0.3606	5.76	0.0561

De acuerdo con los tests de sesgo y kurtosis, los residuales del modelo alcanzan a pasar la prueba de normalidad, considerando una significancia de 0.05 (Tabla 16 y Figura 18). A partir del los resultados del análisis de regresión, la confiabilidad puede ser descrita por el siguiente modelo,

$$\text{conf} = \left(sen\left(-0.06457 + 1.55166\ densidad + 1.217668\ p_{prom} + 0.0049518\ nodos\right)\right)^2 \quad (5.5)$$

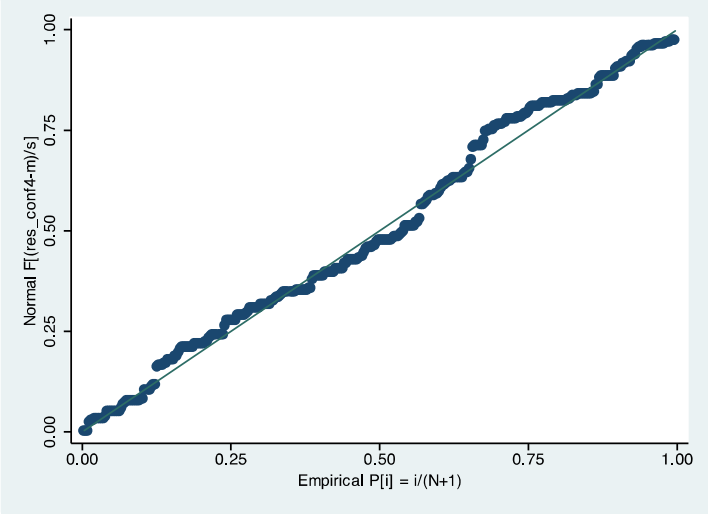

Figura 18. Diagrama de probabilidad normal de los errores de regresión para la confiabilidad

De los resultados obtenidos en esta sección se concluye que las diferencias en la calidad de las soluciones que generan los distintos métodos, son despreciables. Por lo tanto, la cota superior utilizada en la función de aptitud del algoritmo genético (método 3) no conduce a soluciones de menor confiabilidad.

Ahora interesa determinar si existe alguna diferencia significativa en los tiempos de ejecución de los algoritmos.

5.4.2 Análisis de diferencias en los tiempos de corrida

En un análisis preliminar de varianza se definió como variable de respuesta el logaritmo natural del tiempo de ejecución de los algoritmos, y como factores el número de nodos, la densidad de la red y la confiabilidad de los arcos, con sus respectivas interacciones. Después de eliminar los factores poco significativos de este primer análisis, se realizó nuevamente un análisis de varianza y regresión, con los resultados mostrados en la Tabla 17.

Tabla 17. Análisis de regresión de las confiabilidades correspondientes a la mejor solución de cada método

	Number of obs =	324	R-squared	=	0.4771
	Root MSE =	2.08	Adj R-squared =		0.4638

Source	Partial SS	df	MS	F	Prob > F
Model	1243.39308	8	155.424134	35.92	0.0000
Densidad	219.042905	1	219.042905	50.63	0.0000
Nodos	525.969566	1	525.969566	121.57	0.0000
Densidad#Nodos	70.0881753	1	70.0881753	16.20	0.0001
Densidad#Conf_arcos	16.7479175	1	16.7479175	3.87	0.0500
Metodo	49.170228	2	24.585114	5.68	0.0038
Metodo#Densidad	47.6592876	2	23.8296438	5.51	0.0045
Residual	1362.82044	315	4.3264141		
Total	2606.21352	323	8.0687725		

Source	SS	df	MS		
Model	1243.39308	8	155.424134	Number of obs =	324
Residual	1362.82044	315	4.3264141	F(8, 315) =	35.92
Total	2606.21352	323	8.0687725	Prob > F =	0.0000
				R-squared =	0.4771
				Adj R-squared =	0.4638
				Root MSE =	2.08

ln_tiempo	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
Densidad	12.67749	1.618107	7.83	0.000	9.493824	15.86115
Nodos	.1449094	.0131426	11.03	0.000	.1190511	.1707677
c.Densidad# c.Nodos	-.1683407	.0418245	-4.02	0.000	-.2506313	-.08605
c.Densidad# c.Conf_arcos	-2.215359	1.125972	-1.97	0.050	-4.430736	.0000182
Metodo						
2	-.331697	.394277	-0.84	0.401	-1.107446	.4440523
3	-1.280546	.394277	-3.25	0.001	-2.056296	-.504797
Metodo# c.Densidad						
2	-3.29176	1.254735	-2.62	0.009	-5.76048	-.8230392
3	-3.855057	1.254735	-3.07	0.002	-6.323777	-1.386337
_cons	-.5595779	.4828887	-1.16	0.247	-1.509673	.390517

La Ecuación 5.6 relaciona el tiempo de ejecución con los factores significativos en el experimento.

$$\begin{aligned}\ln(tiempo) = {} & -0.5595779 + 12.67749 \cdot densidad + 0.1449094 \cdot nodos \\ & -0.1683407 \cdot densidad \cdot nodos - 2.215359 \cdot densidad \cdot conf\_arcos \\ & - \begin{cases} 0.331697 \text{ si } metodo = 2 \\ 1.280546 \text{ si } metodo = 3 \end{cases} - \begin{cases} 3.291760 \cdot densidad \text{ si } metodo = 2 \\ 3.855057 \cdot densidad \text{ si } metodo = 3 \end{cases}\end{aligned} \tag{5.6}$$

Aunque los residuales de este modelo de regresión no pasaron la prueba de normalidad, los valores del estadístico t mostrados en la Tabla 17 indican que la densidad y el número de nodos son factores

altamente significativos. El valor del estadístico t para el método 3 (algoritmo genético con cotas de confiabilidad) evidencia una diferencia significativa en los tiempos de ejecución con respecto a los otros dos métodos considerados en el modelo. Dado que el $metodo\,3$ y la interacción $densidad \cdot metodo\,3$ tienen los coeficientes más negativos en los términos respectivos de la Ecuación 5.6, se concluye que el método 3 tiende a ser más rápido que los métodos restantes. La ecuación que describe su tiempo de ejecución en términos de la densidad, el número de nodos y la confiabilidad de los arcos es,

$$tiempo = \exp(-1.8401239 + 8.822433 \cdot densidad + 0.1449094 \cdot nodos - 0.1683407 \cdot densidad \cdot nodos - 2.215359 \cdot densidad \cdot conf\_arcos) \qquad (5.7)$$

5.4.3 Pruebas adicionales con el algoritmo de mejor desempeño

Para evaluar el desempeño del método 3 (heurístico AG2) en condiciones más exigentes, se realizaron corridas adicionales en redes de hasta de 200 nodos, con incrementos de 50 nodos y con 5 réplicas para cada tamaño de red. Los tiempos de ejecución se muestran en la Tabla 18 y la Figura 19.

Tabla 18. Tiempos de ejecución del heurístico AG2 en redes de 50 a 200 nodos

Número de nodos	50	100	150	200
	1903	611.5	1848.3	3342.6
	1402	653.9	562	1804.4
	58	9773.1	1657.1	3335.5
	63	581.6	1827.6	1147
	93	979.2	1294.9	5712.6
Mediana	93	653.9	1657.1	3335.5
Media	703.8	2519.86	1437.98	3068.42

De acuerdo con los tiempos registrados, el tiempo de ejecución tiende a crecer polinomialmente con el número de nodos de la red. Con excepción del dato atípico de 9,773.1 segundos obtenido en una instancia con 100 nodos, los tiempos de solución con los diferentes tamaños están por debajo de las dos horas, aún en redes con 200 nodos.

Aunque en la literatura no se han reportado métodos de solución para el problema de asignación de redundancias para maximización de la confiabilidad global de redes generales, investigaciones sobre problemas similares (como el de asignación de redundancias para la maximización de la confiabilidad de la conexión entre un nodo origen y un nodo destino) han podido resolver instancias hasta con 20 nodos (Hernández, 2007).

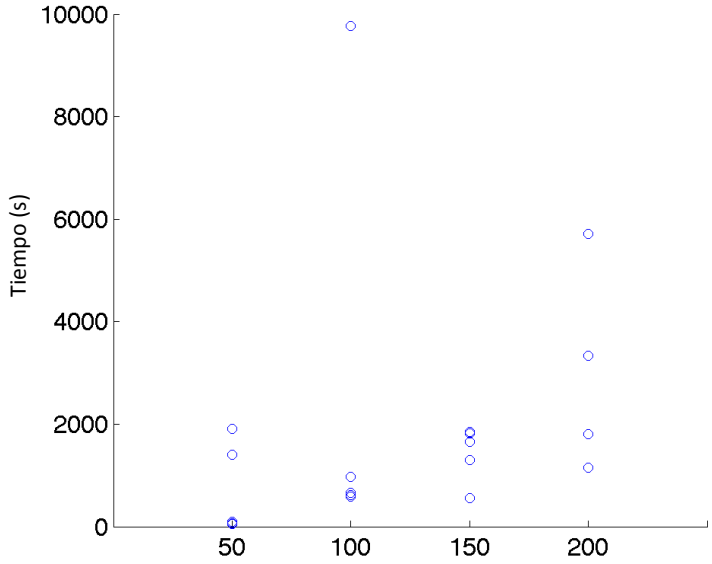

Figura 19. Tiempos de ejecución del algoritmo AG2 en redes de 50 a 200 nodos.

Se presume que la metodología propuesta para la resolución del problema de maximización de confiabilidad global podría adaptarse para resolver problemas como el anteriormente mencionado, o problemas que involucren criterios y restricciones adicionales. Por lo tanto, los algoritmos propuestos en este trabajo constituyen un aporte para la solución de problemas de optimización confiabilidad de redes con dimensiones anteriormente no consideradas en la literatura.

6 Conclusiones

Como se evidenció en este trabajo, la optimización de confiabilidad de redes es un problema doblemente complejo debido a la naturaleza no convexa del problema y a la dificultad para evaluar la confiabilidad. Para poder desarrollar un heurístico eficiente, inicialmente se buscó proponer un método rápido para evaluar la confiabilidad global de redes generales. Con este propósito se estudiaron las condiciones de conectividad de redes y se derivó una ecuación para evaluar la condición suficiente de conectividad a partir de la densidad de la red. Se discutieron y evaluaron varios métodos para determinación de conectividad de redes, de los cuales el método basado en el número de árboles de expansión de la red resultó ser el más rápido. Al combinar este método con las condiciones necesarias y suficientes para determinación de conectividad se obtuvo un algoritmo general para evaluación de conectividad que supera los demás métodos para evaluación de conectividad, considerando diferentes densidades de red.

Posteriormente, el algoritmo para evaluación de conectividad fue utilizado para desarrollar un algoritmo general para evaluación de confiabilidad global de redes. Para lograr una rápida ejecución, este algoritmo inicialmente evalúa la conectividad de la red, luego la reduce y posteriormente selecciona el método más conveniente para la evaluación de confiabilidad, de acuerdo con el número de nodos y el número de árboles de expansión: enumeración completa, inclusión-exclusión con el listado de árboles de expansión de la red, o simulación Monte Carlo con variables antitéticas. Para acelerar los cálculos, se hizo una implementación paralelizada del algoritmo Monte Carlo.

Finalmente se presentaron tres heurísticos para el problema de asignación de arcos en redes, con el propósito de maximizar la confiabilidad global:

- PLES: Algoritmo basado en programación lineal entera secuencial
- AG1: Algoritmo genético que evalúa la confiabilidad mediante el algoritmo general para evaluación de confiabilidad global
- AG2: Algoritmo genético que evalúa la aptitud mediante la cota superior de confiabilidad. Solamente evalúa la confiabilidad en las soluciones que superan la mejor solución de la generación respectiva.

Empíricamente se observó que la cota superior utilizada en este algoritmo funciona adecuadamente como sustituto de la confiabilidad real de la red, cuando la confiabilidad de la red tiende a ser alta.

Al comparar el desempeño de los heurísticos en redes con un máximo de 50 nodos, el heurístico AG2 evidenció menores tiempos de corrida. En todos los escenarios la mediana y el rango de los

tiempos registrados en este heurístico fueron inferiores a los de los demás métodos. La diferencia entre el heurístico AG2 y los demás métodos tiende a ser más notoria en redes de mayor tamaño.

En pruebas adicionales se observó que en redes con 200 nodos el heurístico AG2 puede alcanzar soluciones en menos de 3600 segundos. Se espera que este heurístico pueda dar rápidas soluciones en redes aún mayores. La combinación de técnicas y el uso de cotas de confiabilidad, permiten que este heurístico logre menores tiempos de ejecución que los reportados por trabajos similares de optimización de confiabilidad de redes. Hernández (2007), por ejemplo, había experimentado con redes hasta de 20 nodos, con tiempos de corrida entre 2000 y 4000 segundos.

El heurístico fue concebido para redes generales dirigidas con probabilidades de falla en los arcos, y con nodos totalmente confiables. Sin embargo, mediante la inclusión de arcos ficticios el método se podría extender para redes con nodos con probabilidades de falla. Por otro lado, su extensión para considerar redes dirigidas requeriría modificaciones en la función para evaluación de conectividad.

Investigaciones futuras pueden explorar el desarrollo de métodos para considerar situaciones donde la información sobre costos y confiabilidades es incierta, donde hay correlaciones entre las fallas de los arcos o donde, además de la confiabilidad y los costos, se deben considerar otros elementos en el diseño de la red, tales como el flujo y la capacidad.

7 Bibliografía

Abdesselam, A. (8 de 7 de 2003). Retrieved 8 de 8 de 2009 from arXiv: http://arxiv.org/pdf/math/0306396v2

Aggarwal, K. K., & Rai, S. (1981). Reliability evaluation in computer-communication networks. *IEEE Transactions on Reliability, 30* (1), 32-35.

Altiparmak, F., Dengiz, B., & Smith, A. E. (1998). Reliability optimization of computer communication networks using genetic algorithms. *Proceedings of the IEEE Iinternational Conference on Systems, Man, and Cybernetics* (pp. 4676-4981). San Diego, CA: IEEE.

Ball, M. O. (1986). Computational complexity of network reliability analysis: an overview. *IEEE Transactions on Reliability, 35* (3), 230-239.

Bartlett M. S. (1947) The Use of Transformations. *Biometrics*, 1, 39-52.

Biggs, N. (1974). *Algebraic graph theory.* Cambridge University Press.

Billinton, R., & Allan, R. N. (1992). *Reliability Evaluation of Engineering Systems. Concepts and Techniques* (2nd ed. ed.). New York: Plenum Press.

Billionnet, A. (2008). Redundancy Allocation for Series-Parallel Systems Using Integer Linear Programming. *IEEE Transactions on Reliability, 57* (3), 507-516.

Cancela, H., Rubino, G., & Urquhart, M. E. (2001). An algorithm to compute the all-terminal reliability measure. *Journal of the Indian Operational Research Society (OPSEARCH), 38* (6).

Colbourn, C. J. (1999). Reliability issues in telecommunications network planning. In B. Sanso, & P. Soriano, *Telecommunications Network Planning* (pp. 135-146). Kluwer Academic Press.

Cormen, T., Leiserson, C., & Rivest, R. (1990). *Introduction to Algorithms.* MIT Press and McGraw-Hill.

De Abreu, N. M. (2007). Old and new results on algebraic connectivity of graphs. *Linear Algebra and its Applications, 423*, 53–73.

Deeter, D. L., & Smith, A. E. (1997). Heuristic optimization of network design considering all-terminal reliability. *Proceedingd of the Annual Reliability and Maintainability Symposium.*

Dengiz, B., Altiparmak, F., & Smith, A. (1997). Efficient optimization of all-terminal reliable networks, using an evolutionary approach. *IEEE Transactions on Reliability, 46* (1), 18-26.

Deo, N. (1974). *Graph theory with applications to engineering and computer science* . Englewood Cliffs, N.J.: Prentice Hall.

El Khadiri, M., & Rubino, G. (1992). *A Monte Carlo method based on antithetic variates for network reliability computations. Rapports de Recherche No. 1609.* Institut National de Recherche en Informatique et en Automatique.

El Khadiri, M & Rubino, G (1996) Accelerating the standard Monte Carlo evaluation of highly realiable binary systems. *2nd International Conference on Monte Carlo and Quasi Monte Carlo Methods in Scientic Computing.* - Salzburg, Austria, 1996.

Fard, N., & Lee, T.-H. (2001). Spanning tree approach in all-terminal network reliability expansion. *Computer Communications, 24*, 1348-1353.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal, 23* (2), 298–305.

Gen, M., & Cheng, R. (2000). *Genetic algorithms & engineering optimization* . Jhon Wiley & Sons.

Gen, M., Cheng, R., & Lin, L. (2008). *Network models and optimization multiobjective genetic algorithm approach.* London: Springer-Verlag.

Ha, C., & Kuo, W. (2006). Reliability redundancy allocation: An improved realization. *European Journal of Operational Research, 171*, 24–38.

Ha, C., & Kuo, W. (2006). Reliability redundancy allocation: An improved realization for nonconvex nonlinear programming problems. *European Journal of Operational Research, 171*, 24-38.

Haupt, R. L., & Haupt, S. E. (2004). *Practical genetic algorithms* (2nd edition ed.). Hoboken, New Jersey: John Wiley & Sons.

Hernández, P. A. (2007). *Optimización de redes con arcos de baja confiabilidad adicionando arcos redundantes (Tesis de maestría).* Mayaguez, PR: Departamento de Ingeniería Industrial. Universidad de Puerto Rico.

Jan, R.-H. (1993). Design of reliable networks. *Computers and operations research, 20* (1), 25-34.

Kim, J.-H., & Yum, B.-J. (1993). A heuristic method for solving redundancy optimization problems in complex systems. *IEEE Transactions on realiability, 42* (4), 572-578.

Kiu, S.-w., & McAllister, D. F. (1988). Reliability optimization of computer-communication networks. *IEEE Transactions on Reliability, 37* (5), 475-483.

Knox, L., Christensen, N., & Skordis, C. (2001). The age of the universe and the cosmological constant determined from cosmic microwave background anisotropy measurements. *The Astrophysical Journal, 563*, L95–L98.

Konak, A. (2007). Combining network reductions and simulation to estimate network reliability. In S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, & R. R. Barton (Ed.), *Proceedings of the 2007 Winter Simulation Conference.* (pp. 2301-2305). IEEE.

Konak, A., & Smith, A. E. (1998). A general upper bound for all-terminal network reliability and its uses. *Proceedings of the Industrial Engineering Research Conference.* Banff, Canada: IIE.

Kontoleon, J. M. (1979). Optimum link allocation of fixed topology networks. *IEEE Transactions on Reliability, 28* (2), 145-147.

Kulturel-Konak, S., Smith, A. E., & Coit, D. W. (2003). Efficiently solving the redundancy allocation problem using tabu search. *IIE Transactions, 35*, 515-526.

Kuo, W., & Wan, R. (2007). Recent advances in optimal reliability allocation. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 37* (2), 143-156.

Kuo, W., & Zuo, M. J. (2003). *Optimal reliability modeling: principles and applications.* Hoboken, NJ: Jhon Wiley & Sons.

Kuo, W., Pradad, V. R., Tillman, F. A., & Hwang, C.-L. (2001). *Optimal reliability design: fundamentals and applications.* Cambridge: Cambridge University Press.

Lemieux, C. (2009). *Monte Carlo and Quasi-Monte Carlo Sampling* . Springer.

Lewis, P. A., & Orav, E. (1989). *Simulation methodology for statisticians, operations analysts, and engineers.* CRC Press.

Liang, Y.-C., & Chen, Y.-C. (2007). Redundancy allocation of series-parallel systems using a variable neighborhood search algorithm. *Reliability Engineering and System Safety, 92*, 323-331.

Liu, C., Dai, M., & Wu, X.-Y. (1999). A new algorithm for computing the overall network reliability. *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, VI*, 149-152.

Melanie, M. (1996). *An introduction to genetic algorithms.* Cambridge, Massachusetts: MIT Press.

Papadimitriou, C.H., Steiglitz, K. (1998) *Combinatorial optimization: Algorithms and Complexity*. Dover Publications.

Ramirez-Marquez, J. E., Coit, D. W., & Konak, A. (2004). Redundancy allocation for series-parallel systems using a max-min approach. *IIE Transactions, 36*, 891-898.

Rubino, G. (1998). Network Reliability Evaluation. In K. Bagchi, & J. Walrand, *State-of-the art in performance modeling and simulation* (pp. 275-302). Gordon and Breach Books.

Satyanarayana, A., & Hagstrom, J. N. (1981). A new algorithm for the reliability analysis of multi-terminal networks. *IEEE Transactions on Reliability, 30* (4), 325-333.

Sharafat, A. R., & Ma'rouzi, O. R. (2009). All-terminal network reliability using recursive truncation algorithm. *IEEE Transactions on Reliability, 58* (2).

Shi, D. H. (1987). A new heuristic for constrained redundancy-optimization in complex systems. *IEEE Transactions on Reliability, 36* (5), 621-623.

Shooman, M. L. (2002). *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design.* New York: John Wiley & Sons.

Spielman, D. A. (2007). Spectral graph theory and its applications. *48th Annual IEEE Symposium on Foundations of Computer Science* (pp. 29-38). Providence, RI, USA: IEEE.

Sung, C. S., Cho, Y. K., & Song, S. H. (2003). Combinatorial Reliability Optimization. In H. Pham, *Handbook of reliability engineering* (pp. 91-114). London: Springer.

Tavakkoli-Moghaddam, R., Safari, J., & Sassani, F. (2008). Reliability optimization of series-parallel systems with a choice of redundancy strategies using a genetic algorithm. *Reliability Engineering and System Safety, 93*, 550-556.

Wilkov, R. S. (1972). Analysis and design of reliable computer networks. *IEEE Transactions on Communications, 20* (3), 660-678.

Yalaoui, A., Châtelet, E., & Chu, C. (2005). A new dynamic programming method for reliability & redundancy allocation in a parallel-series system. *IEEE Transactions on Reliability, 54*, 254-261.

8 Apéndices

Apéndice 1

Programa para comparar los tiempos de ejecución de seis métodos para determinación de conectividad

```
%% Función que compara los tiempos de ejecución de 6 métodos para determinar la conectividad de redes generales

function[tiempo]=Comparacion_metodos_conectividad2(N,n,p)
% N : Número de réplicas
% n : Número de nodos
% p : Confiabilidad de las conexiones

% Se inicializan las matrices a utilizar
tiempo=zeros(6,n);
conexion=zeros(6,n,N);

for g=2:n % Contador del numero de nodos
        for h=1:N % Número de réplicas
        % Generación aleatoria de la matriz simétrica de adyacencia
                C=triu(rand(g)<p,1);
                C=C+C';

%% Existencia de un arbol de expansión
%Matriz de adyacencia del grafo
    A=triu(C,1)+tril(C,-1);
    tic;
% Conjunto de nodos de la red
    Nodos=1:g;
% Índices del primer valor diferente de cero
    [fila columna]=find(A,1);
% Actualización de los conjuntos de nodos conectados y no conectados
    Conectados=union(fila,columna);
    No_conectados=setdiff(Nodos,Conectados);
if isempty(Conectados)
      conexion(1,g,h)=0;
else
      conexion(1,g,h)=1;
for i=2:(g-1)
% Índices de la primera posición diferente de cero
        [fila columna]=find(A(No_conectados,Conectados),1);
% Se verifica si "fila" es con conjunto vacio
if isempty(fila)
            conexion(1,g,h)=0;
break;
end
% Actualización de la información de los conjuntos
        Conectados=union(No_conectados(fila),Conectados);
        No_conectados=setdiff(Nodos,Conectados);
end
end
    tiempo(1,g)=tiempo(1,g)+toc;
```

```
%% Ruta mas corta (Algoritmo de Floyd)
      L2=diag(ones(1,g))+C; % Matriz de conexión
      D=1./L2; % Matriz de distancias
      tic % Se inicia el cronómetro
for j=1:g
for i=1:g
for k=1:g
if ((i~=k) && (j~=k) && (i~=j))
if (D(i,j)<inf)&& (D(j,k)<inf)
                  D(i,k)=min(D(i,j)+D(j,k),D(i,k));
end
end
end
end
end
      ruta_total=sum(D(:));
conexion(2,g,h)=(ruta_total<inf);
      tiempo(2,g)=tiempo(2,g)+toc; % Se para el cronómetro

%% Conectividad Algebraica
      tic % Se inicia el cronómetro
      L=diag(sum(C))-C; % Matriz laplaciana
      eigenvalor=sort(eig(L));
      conexion(3,g,h)=eigenvalor(2)>0.01;
      tiempo(3,g)=tiempo(3,g)+toc; % Se para el cronómetro

%% Número de arboles de expansión
      tic % Se inicia el cronómetro
      L=diag(sum(C))-C; % Nuevamente se obtiene la Matriz Laplaciana para
% hacer la comparación en igualdad de condiciones
conexion(4,g,h)=(det(L((1:g-1),(1:g-1))))>0;
tiempo(4,g)=tiempo(4,g)+toc; % Se para el cronómetro

%% Multiplicación de la matriz de conexión
      tic % Se inicia el cronómetro
      D=diag(ones(1,g))+C;
E=D*D;
      E=(E>0)*1;
while (any(any(triu(D~=E,1)))&&any(any(triu(E==0,1))))
D=E;
        E=D*D;
        E=(E>0)*1;
end
conexion(5,g,h)=1-1*any(any(triu(E==0,1)));
tiempo(5,g)=tiempo(5,g)+toc; % Se para el cronómetro

%% Potencia de la matriz de conexion
      tic % Se inicia el cronómetro
      D=diag(ones(1,g))+C;
E=D^(g-1);
      conexion(6,g,h)=1-1*any(any(triu(E==0,1)));
tiempo(6,g)=tiempo(6,g)+toc; % Se para el cronómetro
end
```

```
end
tiempo=tiempo./N;

%% Creación de la figura
figure1 = figure('Color',[1 1 1]);
axes1 = axes('Parent',figure1,'Position',[0.13 0.0926 0.775 0.7822]);
box('on');
hold('all');
plot1 = plot(tiempo','Parent',axes1);
set(plot1(1),'Marker','o','DisplayName','Existencia de árbol de expansión');
set(plot1(2),'Marker','square','DisplayName','Rutas más cortas');
set(plot1(3),'DisplayName','Conectividad algebraica');
set(plot1(4),'DisplayName','Número de arboles de expansión');
set(plot1(5),...
'DisplayName','Multiplicación booleana de la matriz de conexión');
set(plot1(6),'DisplayName','Potencia de la matriz de conexión');
xlabel('Número de nodos','FontSize',11,'FontName','Calibri');
ylabel('Tiempo promedio (segundos)','FontSize',11,'FontName','Calibri');
title({'Tiempos de ejecución de los métodos para determinación de conectividad','Considerando diferentes tamaños de red'},...
'FontWeight','bold',...
'FontSize',12,...
'FontName','Calibri');
legend1 = legend(axes1,'show');
set(legend1,'Position',[0.1492 0.5564 0.3738 0.2839]);
```

Apéndice 2

Función para la evaluación de conectividad de redes generales

```
% Función que evalúa la conectividad de redes generales a partir de la
% matrix de adyacencia

function[conexion]=conectividad(Matriz_Adyacencia,grado)
% grado: Es la dimensión de Matriz_Adyacencia

conexion=0;
suma_grado=sum(Matriz_Adyacencia(:));
if suma_grado<grado^2-3*grado+4
if suma_grado>=2*(grado-1)
if all(sum(Matriz_Adyacencia)>=1)
        L=diag(sum(Matriz_Adyacencia))-Matriz_Adyacencia;
        conexion=(det(L((1:grado-1),(1:grado-1))))>0;
end
end
else
   conexion=1;
end
end
```

Apéndice 3

Función para la evaluación de confiabilidad mediante la enumeración completa de los estados de la red

```
% Función que calcula la confiabilidad exacta de una red mediante
% enumeración completa

function R=enumeracion_completa(Matriz)

% Numero de nodos de la red
n=length(Matriz);

% Matriz triangular superior de unos
b = triu(ones(n),1);

% Número de conexiones de la red
m=n*(n-1)/2;

% Se obtiene la matriz de diseño factorial completo 2^n. Cada fila
% representa uno de los posibles estados de los componentes.
B=factorial2n(m);

% Número de filas de la matriz de diseño
a=2^m;

%% Las variables hasta esta linea se pueden declarar como globales

%% Obtención del vector de confiabilidades, de acuerdo con el valor de X

% Se obtiene un vector con las confiabildiades, mediante la función
% vector_confiabilidad
% Confiabilidades=vector_confiabilidad(Matriz);
% Confiabilidades=single([0.2425    0.9358    0.8729    0.9945    0.1410         0]);

Confiabilidades=Matriz(b==1);
conexion=zeros(a,1);

% Determinación de cantidad de árboles y conexiones

for i=1:a
   C=b;
% Matriz simétrica de adyacencia
   C(b==1)=B(i,:);
   C=C+C';
% Se obtiene la Matriz Laplaciana
   conexion(i)=conectividad(C,n);
end

r=bsxfun(@times,(B==0),(1-Confiabilidades'))+bsxfun(@times,(B==1),Confiabilidades');
% Cada fila de la matriz de estados B se multiplica por el vector de
% probabilidades, y por su complemento

R=conexion'*prod(r,2);  %% Confiabilidad exacta de la red

end
```

Apéndice 4

Programa para el cálculo de la confiabilidad global, mediante el listado de árboles de expansión de la red

```
%% Estimación de la confiabilidad exacta para una red dirigida
% Mediante el listado de árboles de expansión de la red

function[confiabilidad]=inclusión_exclusion(Matriz_conf)
% Matriz_conf: es una matriz con los valores de las confiabilidades de los
% arcos

Matriz=1*(Matriz_conf>0);
Red=Matriz(triu(true(size(Matriz)),1))';
Arboles=All_spanning_trees(Matriz);

%% Organización de los datos
m=length(Red); % Longitud de los vectores
N_trees=size(Arboles,1); % Numero de árboles
confiabilidades=Matriz_conf(triu(true(size(Matriz)),1))'; % Vector con las confiabilidades

%% Obtención de la función de estructura
a=ff2n(N_trees); % Matriz factorial
a(1,:)=[];
filas=(2^N_trees)-1; % Numero de filas de la matriz factorial resultante
% Corresponde al número de términos de la función de estructura

% Cálculo intermedio para la función de estructura
C=zeros(filas,m);
for i=1:filas
for j=1:N_trees
      C(i,:)=a(i,j).*Arboles(j,:)+C(i,:);
end
end

C=(C>0);% Representación matricial de la función de estructura

%% Cálculo de la confiabilidad del sistema
x=zeros(filas,1);
for i=1:filas
c=nonzeros(C(i,:).*confiabilidades);
x(i)=prod(c).*((-1)^(sum(a(i,:))+1));
end
confiabilidad=sum(x);
```

Apéndice 5

Programa para generar el listado de árboles de expansión de una red

```
%% Función que determina todos los arboles de expansión de un grafo no
%% dirigido

function [Matriz_arboles]=All_spanning_trees(Matriz)
% Matriz : es la matriz de adyacencia
% Matriz_arboles: es una matriz en la cual cada una de las filas representa
% uno de los árboles de expansión del grafo

%% Cálculos preliminares

% Se determina el número de nodos de la matriz
n=size(Matriz,1);

% Se crea la matriz B, en la cual a cada conexión se le asigna un número
% que corresponde al índice unidimensional de su posición en la matriz
% respectiva
B=triu(Matriz,1);
index=find(B);
B(index)=index;
B=B+B';

% Para obtener n-1 cortes, se suprime la fila que tiene mayor número de
% elementos
[value index]=max(sum(Matriz,2));
B(index,:)=[];


%% Se obtiene el producto cartesiano de los n-1 cortes de la matriz

% Para esto se obtienen las combinaciones de los elementos de cada uno de
% los cortes.
Combinaciones = setprod2(B)';

% Se eliminan elementos que no hacen parte del conjunto de árboles de
% expansión (columnas con elementos repetidos).
Arboles=Combinaciones(:,logical(prod(diff(sort(Combinaciones)))));

% Se organizan los elementos de cada columna
Arboles=sort(Arboles);
% Se organizan las columnas de la matriz (columnas similares pasan a ocupar
% posiciones adyacentes
Arboles= sortrows(Arboles')';

% Se obtienen las diferencias entre columnas. Cero en todos los elementos
% de la columna i, indica que las columnas i e i+1 son idénticos.
Indices=~any(diff(Arboles')');

% A partir de los índices, se obtienen índices acumulados (cuentan el
% número de veces que se repite cada columna de la matriz
if(numel(Indices)>0)
   Acumulado=ones(1,length(Indices));
   Acumulado(1)=Indices(1);
for i=2:length(Indices)
if (Indices(i)==1) && (Indices(i-1)==1)
```

```
        Acumulado(i)=1+Acumulado(i-1);
else
if (Indices(i)==1) && (Indices(i-1)==0)
            Acumulado(i)=1;
end
end
end

   Resultado=[Indices.*Acumulado,0];
   Index=false(1,length(Arboles));

% Se identifican las columnas que están repetidas un número par de veces
for i=1:(length(Resultado)-1)
if Resultado(i)>Resultado(i+1)
if mod(Resultado(i)+1,2)==0
            Index(:,(i-Resultado(i)+1):(i+1))=true;
end
end
end
if Resultado(end)==1
Index(:,(end-1):end)=true;
end

% De acuerdo con los índices obtenidos, se suprimen las columnas que
% no son árboles de expansión
   Arboles(:,Index)=[];

% También se eliminan columnas repetidas
   Arboles=unique(Arboles','rows')';

% La matriz obtenida se convierte en una matriz binaria, en la que cada
% columna representa una conexión, y cada fila es un árbol de expansión
% de la red.
   N_arboles=size(Arboles,2);
   N_elementos=n*(n-1)/2;
   Matriz_arboles=zeros(N_arboles,N_elementos);
b=triu(ones(n),1);
   Vector_indices=find(b);
for i=1:N_arboles
      Matriz_arboles(i,:)=ismember(Vector_indices,Arboles(:,i));
end

else
Matriz_arboles=[];
end


% Función para generar el producto cartesinano de los cortes de la red
% Adaptación de la función SETPROD, desarrollada por Mukhtar Ullah.
% La función original está disponible en MatlabCentral:
% http://www.mathworks.com/matlabcentral/fileexchange/5898.
% El argumento de la función modificada es una matriz, en la que cada fila
% corresponde a uno de los n-1 cortes del grafo, y cada columna
% representa uno de los nodos de la red.

function C = setprod2(Matriz)

filas=size(Matriz,1);

args=cell(1,filas);

for i=1:filas
```

```
    args{i}=nonzeros(Matriz(i,:));
end

[F{1:filas}] = ndgrid(args{:});

for i=filas:-1:1
    G(:,i) = F{i}(:);
end

C = unique(G , 'rows');
```

Apéndice 6

Programa para evaluación de confiabilidad mediante Monte Carlo con variables antitéticas

```
% Función que estima la confiabilidad de una red por el método Monte Carlo
% con variables antitéticas

function [confiabilidad varianza]=montecarlo(Matriz,rep)

% rep: Número de réplicas de la simulación
rep=round(rep/2);
n=length(Matriz);
b = triu(ones(n),1);
vector_confiabilidades=Matriz(b==1);
m=length(vector_confiabilidades);

conexion=0;
for i=1:rep
   aleatorios=rand(m,1);
C1=b;
   C1(b==1)=(aleatorios<vector_confiabilidades);
   C1=C1+C1';
   C2=b;
   C2(b==1)=((1-aleatorios)<vector_confiabilidades);
   C2=C2+C2';
   Qk=(conectividad(C2,n)+conectividad(C1,n))/2;
   conexion=Qk+conexion;
   V=V+Qk^2;
end
confiabilidad(1)=conexion/rep;
varianza=V/(rep*(rep-1))+confiabilidad^2/(rep-1);
end
```

Apéndice 7

Función para determinar la cota superior de confiabilidad global en redes no dirigidas

```
% Función que estima la cota superior de la confiabilidad de una red
% mediante el método de Konak y Smith

% An Improved General Upper Bound for
% All-Terminal Network Reliability
% Abdullah Konak and Alice E. Smith
% University of Pittsburgh

function [cota]=cota_superior(matriz_conf)

% Argumento de la función: Matriz simétrica de Confiabilidades

n=length(matriz_conf);
suma=prod(1-matriz_conf(:,1));
for i=2:n
  j=i-1;
termino2=(1-prod(1-matriz_conf(:,j))./(1-matriz_conf(i,j)));
suma=prod(1-matriz_conf(:,i))*prod(termino2)+suma;
end
cota=1-suma;

end
```

Apéndice 8

Programa para la reducción de redes generales

```
% Función que realiza reducciones en paralelo, reducciones grado 1 y
% reducciones grado 2

function [R_red lambda]=algoritmo_reduccion(Rx,X,Re,E)

% Rx: Confiabilidad de los nuevos arcos
% X: Número de nuevos arcos
% Re: Confiabilidad de los arcos existentes
% E: Número de arcos existentes

% Reducciones de arcos en paralelo
R=1-((1-Re).^E).*((1-Rx).^X);
R_red=R;
lambda=1;
repite1=true;
repite2=true;
while repite1||repite2
% Reducciones de grado 1
nodos1=find(sum(R_red>0)==1);
if any(nodos1)
[variable variable conf]=find(R_red(nodos1(1),:));
      R_red(nodos1(1),:)=[];
      R_red(:,nodos1(1))=[];
      lambda=lambda*prod(conf);
repite1=true;
else
      repite1=false;
end
% Reducciones de grado 2
nodos2=find(sum(R_red>0,2)==2);
if any(nodos2)
fila=R_red(:,nodos2(1));
      [filas columnas conf]=find(fila);
z=sum(conf)-prod(conf);
R_red(filas(1),filas(2))=1-(1-R_red(filas(1),filas(2)))*(1-prod(conf)/z);
      R_red(filas(2),filas(1))=R_red(filas(1),filas(2));
      R_red(nodos2(1),:)=[];
      R_red(:,nodos2(1))=[];
lambda=lambda*z;
      repite1=true;
else
      repite2=false;
end
end
end
```

Apéndice 9

Programa para la evaluación de confiabilidad de redes generales

```
% Función general para la evaluación de confiabilidad de redes generales

function [R]=evaluacion_general_confiabilidad(Rx,X,Re,E,replicas)

% Rx: Es la confiabilidad de los nuevos arcos
% Re: Es la confiabilidad de los arcos existentes
% X : Es el número de nuevos arcos
% E : Es el número de arcos existentes

C=(X+E)>0;
grado=size(C,1);
if conectividad(C,grado)==1
   [Red lambda]=algoritmo_reduccion(Rx,X,Re,E);
n=size(Red,1);
   m=n*(n-1)/2;
if m==0
      R=lambda;
else
if m<=15
R=enumeracion_completa(Red);
else
         T=numero_arboles(Red);
if T>10
            R=montecarlo(Red,replicas);
else
            all_spanning_trees(Red);
            R=inclusion_exclusion(Red);
end
end
      R=lambda*R;
end
else
   R=0;
end
```

Apéndice 10

Programa para optimización de confiabilidad global mediante programación lineal entera secuencial

```
function[solucion2]=prog_lin_sec()

% A: Vector de costos de los arcos en cada conexión;

% LB: Cota inferior del número de arcos en cada conexión;

% UB: Cota superior del número de arcos en cada conexión;

% n           :   Número de nodos de la red


% existentes    :   Matriz nxn con el número de arcos preexistentes en cada
%                conexión


% conf_exist    :   Matriz simétrica de confiabilidades de los arcos
%                preexistentes

% conf_nuevos   :   Matriz simétrica de confiabilidadees de los nuevos
%                arcos

% presupuesto   :   Recursos totales disponibles para asignar a los nuevos
%                arcos



load datos_problema
replicas=10000;
tic;

max_iteraciones=50;
max_repetidas=5;

conteo_iteraciones=1;
conteo_sol_rep=0;

subplot(2,1,1); ylabel('Costo'); ylim([0 presupuesto]);

subplot(2,1,2);ylabel('Confiabilidad');

% Matriz con la adición de un elemento en la diagonal principal
suma_diagonal=repmat(LB,nvars,1)+eye(nvars);

% Matriz inicial
matriz_inicial=[LB;suma_diagonal;UB];

% Número de filas de la matriz inicial
Filas=nvars+2;

% Se incluyen los arcos preexistentes
matriz_regresion=matriz_inicial+repmat(existentes(b)',Filas,1);
```

```
% Se permite como máximo la adición de un arco en cada conexión, siempre
% que no exceda el límite superior
Limite_sup=min(LB+1,UB);

solucion=regresion_optimizacion(matriz_regresion,Limite_sup,LB,Filas,true);

% Impresión de tabla de resultados
confiabilidad=zeros(50,1);
costo_total=zeros(50,1);

b=triu(true(n),1);
matriz_solucion=zeros(n);
matriz_solucion(b)=solucion;
matriz_solucion=matriz_solucion+matriz_solucion';
confiabilidad(1)=evaluacion_general_confiabilidad(conf_nuevos,matriz_solucion,conf_exist,existentes,replicas);
costo_total(1)=solucion*A';
% soluciones=zeros(1,nvars);

tiempo(1)=toc;
fprintf('Iteracion \tTiempo \t\tConf \t\tCosto \n')
fprintf('\t%g \t\t%.2f  \t\t%.5f \t\t%g \n',conteo_iteraciones,tiempo,confiabilidad(1),costo_total(1));

while (conteo_iteraciones<max_iteraciones) && (conteo_sol_rep<max_repetidas)

% Matriz con la adición de un elemento en la diagonal principal
   suma_diagonal=repmat(solucion,nvars,1)+eye(nvars);

% Se eliminan las filas que exceden el límite superior de la cantidad de
% arcos en cada conexión
   suma_diagonal(any(suma_diagonal>repmat(UB,nvars,1),2),:)=[];

% Matriz con la sustracción de un elemento en la diagonal principal
   resta_diagonal=repmat(solucion,nvars,1)-eye(nvars);

% Se eliminan las filas que que violan la cantidad mínima de arcos
   resta_diagonal(any(resta_diagonal<repmat(LB,nvars,1),2),:)=[];

% Matriz de regresión
   matriz_regresion=[solucion;resta_diagonal;suma_diagonal;UB];

% Número de filas de la matriz de regresión
   Filas=size(matriz_regresion,1);

% Se incluyen los arcos preexistentes
   matriz_regresion=matriz_regresion+repmat(existentes(b)',Filas,1);

% Se permite como máximo la adición de un arco en cada conexión, siempre
% que no exceda el límite superior
   Limite_sup=min(solucion+1,UB);

% Se permite la sustracción de un arco en cada conexión, siempre
% que no exceda el límite inferior
   Limite_inf=max(solucion-1,LB);

   solucion2=regresion_optimizacion(matriz_regresion,Limite_sup,Limite_inf,Filas,false);
   soluciones(conteo_iteraciones,:)=solucion2;
   conteo_iteraciones=conteo_iteraciones+1;

if solucion==solucion2
      conteo_sol_rep=conteo_sol_rep+1;
end
   solucion=solucion2;
```

```
    matriz_solucion=zeros(n);
    matriz_solucion(b)=solucion;
    matriz_solucion=matriz_solucion+matriz_solucion';

confiabilidad(conteo_iteraciones)=evaluacion_general_confiabilidad(conf_nuevos,matriz_solucion,conf_exist,existentes,repli
cas);
    costo_total(conteo_iteraciones)=solucion*A';

    tiempo(1)=toc;
    fprintf('\t%g\t\t%.2f                                                                    \t\t%.5f
\t\t%g\n',conteo_iteraciones,tiempo,confiabilidad(conteo_iteraciones),costo_total(conteo_iteraciones));

% Grafica de resultados
hold all;
    subplot(2,1,1);
    plot(costo_total(1:conteo_iteraciones),'r');
subplot(2,1,2);
    plot(confiabilidad(1:conteo_iteraciones),'b');
figure(gcf);
end
display(soluciones);
end

function solucion=regresion_optimizacion(matriz_regresion,Limite_sup,Limite_inf,Filas,inicial)
% A: Vector de costos

load datos_problema

% Se inicializa vector de confiabilidades
Confiabilidades=zeros(Filas,1);
replicas=10000;

% Calculo de confiabilidades
for i=1:Filas
    X=zeros(n);
X(b)=matriz_regresion(i,:);
    X=X+X';
    Confiabilidades(i)=-1*evaluacion_general_confiabilidad(conf_nuevos,X,conf_exist,existentes,replicas);
end

if inicial==false

% Indice de las columnas que tienen todos sus elementos repetidos
    iguales=all(matriz_regresion==repmat(min(matriz_regresion),Filas,1));

% Se suprimen las filas repetidas
    matriz_regresion(:,iguales)=[];

end

% Se agrega una columna de unos para el término independiente del modelo de
% regresión
matriz_regresion=[ones(Filas,1),matriz_regresion];

% Se obtienen los estimadores del modelo de regresión lineal
betas=regress(Confiabilidades,matriz_regresion);

if inicial==false
% Se inicializa el vector betas2
    betas2=zeros(nvars,1);
```

```
% Se asignan valores de cero en las columnas con elementos repetidos
   betas2(iguales)=0;
   betas2(~iguales)=betas(2:end);
else
   betas2=betas(2:end);
end

% Se resuelve el modelo de programación lineal, mediante el método simplex
options = optimset('Display','off','LargeScale', 'off', 'Simplex', 'on');
solucion=linprog(betas2,A,presupuesto,[],[],Limite_inf,Limite_sup,[],options)';
end
```

Apéndice 11

Algoritmo genético para optimización de confiabilidad global

```
function []=alegoritmo_genetico_propio()

% global A LB UB b conf_exist conf_nuevos costo existentes max_red maximo minimo n nvars pop_size presupuesto

%% Notación
% A: Vector de costos de los arcos en cada conexión;

% LB: Cota inferior del número de arcos en cada conexión;

% UB: Cota superior del número de arcos en cada conexión;

% n            :   Número de nodos de la red

% existentes    :   Matriz nxn con el número de arcos preexistentes en cada
%                  conexión

% conf_exist    :   Matriz simétrica de confiabilidades de los arcos
%                  preexistentes

% conf_nuevos   :   Matriz simétrica de confiabilidadees de los nuevos
%                  arcos

% presupuesto   :   Recursos totales disponibles para asignar a los nuevos
%                  arcos

% pop_size      :   Tamaño de la población;

tic;

% Se cargan los datos del problema
load datos_problema;

n=size(existentes,1);
b=triu(true(n),1);

% Parametros del algoritmo genetico
pop_size=20;
max_generaciones=20;
max_estancamiento=10;
frac_cruce=0.7;
n_torneo=4;
n_elite=2;
tolerancia=1e-5;
n_nodos=size(existentes,1);
escala=0.5;
contraccion=1;

n_descendencia=round(frac_cruce*pop_size);
n_mutacion=pop_size-n_descendencia-n_elite;
long_genoma=n*(n-1)/2;
replicas=10000;

%  Población inicial
[poblacion fitness_values mejor promedio]=
poblacion_inicial(b,A,long_genoma,n,presupuesto,n_nodos,pop_size,UB,LB,conf_exist,existentes,conf_nuevos,replicas);
```

```
% Variable que representa la mejor solución actual

% Variable que contiene los valores de la función de aptitud
descendencia=zeros(n_descendencia,long_genoma);
n_generaciones=1;
estancamiento=0;
mejores=zeros(max_generaciones,1);
promedio_poblacion=zeros(max_generaciones,1);
desv_est_poblacion=zeros(max_generaciones,1);
solucion=zeros(max_generaciones,long_genoma);
tiempo=zeros(max_generaciones,1);

% Inicializa grafico
hold on;
set(gca,'xlim',[0,max_generaciones]);
set(gca,'ylim',[0,1.1]);
xlabel('Generacion','interp','none');
ylabel('Fitness value','interp','none');
plotMejor = plot(1,mejor,'.k');
set(plotMejor ,'Tag','Mejor solucion');
plotPromedio = plot(1,promedio,'.b');
set(plotPromedio,'Tag','Aptitud promedio');
title(['Mejor: ',' Promedio: '],'interp','none')
shg;

% Inicializa tabla
fprintf('\n Algoritmo Genético \n')
fprintf('\n Iteracion \tTiempo \t\tConf. Prom \t\tConf. Mejor \tCosto \n')

while (n_generaciones<=max_generaciones) && (estancamiento<max_estancamiento)

% Se selecciona la elite
   poblacion_ordenada=sortrows([fitness_values,poblacion],1);
   poblacion_ordenada(:,1)=[];
   elite=poblacion_ordenada(1:n_elite,:);

% Se obtiene la descendencia

for i=1:n_descendencia

descendencia(i,:)=cruce_genetico(b,A,n,presupuesto,existentes,poblacion,long_genoma,pop_size,n_torneo,fitness_values);
end

% Se obtienen las mutaciones
   mutaciones=mutacion_factible(b,A,n,presupuesto,existentes,long_genoma,UB
,LB,n_generaciones,max_generaciones,n_mutacion,poblacion,escala,contraccion);

% Se conforma la nueva poblacion
   poblacion=[elite;descendencia;mutaciones];

% Se confabiliza el estancamiento
if (n_generaciones>=2)
if (mejores(n_generaciones)-mejores(n_generaciones-1))<tolerancia
        estancamiento=estancamiento+1;
else
        estancamiento=0;
end
end

   fitness_values=funcion_aptitud(n,pop_size,conf_exist,existentes,conf_nuevos,poblacion,mejor);
   [mejores(n_generaciones) ind_mejor]=max(fitness_values);
   promedio_poblacion(n_generaciones)=mean(fitness_values);
```

```
    desv_est_poblacion(n_generaciones)=std(fitness_values);
    mejor=mejores(n_generaciones);

% Se actualiza la cuenta del número de generaciones
    promedio=promedio_poblacion(n_generaciones);

% Se actualiza el grafico
    nuevoX = [get(plotMejor,'Xdata') n_generaciones];
    nuevoY = [get(plotMejor,'Ydata') mejor];
    set(plotMejor,'Xdata',nuevoX, 'Ydata',nuevoY);
    nuevoY = [get(plotPromedio,'Ydata') promedio];
    set(plotPromedio ,'Xdata',nuevoX, 'Ydata',nuevoY);
set(get(gca,'Title'),'String',['Mejor: ',num2str(mejor),' Promedio: ',num2str(promedio)]);
shg;

    tiempo(n_generaciones)=toc;

    solucion(n_generaciones,:)=poblacion(ind_mejor,:);
    confiabilidad=mejores(n_generaciones);
    costo=A*solucion(n_generaciones,:)';

% Imprime tabla
    fprintf('\t%g \t\t%.2f  \t\t%.5f \t\t%.5f \t\t%g \n',n_generaciones,tiempo(n_generaciones),promedio,confiabilidad,costo);

    n_generaciones=n_generaciones+1;

end
    solucion(n_generaciones:end,:)=[];
    display(solucion);
    fprintf('La configuracion optima es: \n');
    display(solucion(n_generaciones-1,:));
    LegnD = legend('Mejor solucion','Aptitud promedio');
set(LegnD,'FontSize',8);
    hold off;

end


function [matriz_nuevos]=decodificacion(n,vector_nuevos)

% nuevos : Vector con solución codificada

% Se decodifica la solución
b=triu(true(n),1);
matriz_nuevos=zeros(n);
matriz_nuevos(b)=vector_nuevos;
matriz_nuevos=matriz_nuevos+matriz_nuevos';
% Se calcula la matriz de confiabilidades
end

function [inicial cotas mejor promedio]=
poblacion_inicial(b,A,long_genoma,n,presupuesto,n_nodos,pop_size,UB,LB,conf_exist,existentes,conf_nuevos,replicas)

% Se genera una población de dos veces el tamaño de la población requerida
muestra=2*pop_size;
inicial=generar_poblacion(b,A,n,presupuesto,existentes,muestra,UB,LB,long_genoma);

% Se suprimen los elementos repetidos de la población
inicial=unique(inicial,'rows');

% Si el tamaño de la población resultante es menor que la población
% requerida, se genera aleatoriamente la cantidad de elementos faltantes
```

```
if size(inicial,1)<pop_size
   muestra=pop_size-size(inicial,1);
faltantes=generar_poblacion(b,A,n,presupuesto,existentes,muestra,UB,LB,long_genoma);
   inicial=[inicial;faltantes];
else
% De lo contrario, si el tamaño de la población resultante es mayor que la
% población requerida, se suprimen los elementos con menor valor de la cota
% superior de probabilidad
if size(inicial,1)>pop_size
      filas=size(inicial,1);
cotas=zeros(filas,1);
for i=1:filas
         matriz_nuevos=decodificacion(n_nodos,inicial(i,:));
         matriz_conf=1-((1-conf_exist).^existentes).*((1-conf_nuevos).^(matriz_nuevos));
         cotas(i)=cota_superior(matriz_conf);
end
      inicial=sortrows([cotas,inicial],1);
      inicial(1:(filas-pop_size),:)=[];
end
[mayor_cota indice]=max(inicial(:,1));
   matriz_nuevos=decodificacion(n_nodos,inicial(indice,2:end));
   mejor=evaluacion_general_confiabilidad(conf_nuevos,matriz_nuevos,conf_exist,existentes,replicas);
   cotas=inicial(:,1);
   cotas(indice)=mejor;
   inicial(:,1)=[];
   promedio=mean(cotas);
end

end


% Función que de manera aleatoria genera una población de tamaño muestra
function [inicial]=generar_poblacion(b,A,n,presupuesto,existentes,muestra,UB,LB,long_genoma)
inicial=zeros(muestra,long_genoma);
i=1;
while i<=muestra
% Se genera aleatoriamente una red
vector=floor(rand(1,long_genoma).*(UB-LB+1))+LB;
if factible(A,n,presupuesto,existentes,vector,b)
inicial(i,:)=vector;
      i=i+1;
end
end
end


function [fitness_values]=funcion_aptitud(n,pop_size,conf_exist,existentes,conf_nuevos,poblacion,mejor)

replicas=10000;
fitness_values=zeros(pop_size,1);
for i=1:pop_size
   x=poblacion(i,:);
matriz_nuevos=decodificacion(n,x);
% matriz_conf=1-((1-conf_exist).^existentes).*((1-conf_nuevos).^(matriz_nuevos));
% fitness_values(i)=cota_superior(matriz_conf);
% if fitness_values(i)>mejor
      fitness_values(i)=evaluacion_general_confiabilidad(conf_nuevos,matriz_nuevos,conf_exist,existentes,replicas);
% end
end

end
```

```
function [padres]=seleccion_padres(long_genoma,pop_size,n_torneo,poblacion,fitness_values)

padres=zeros(2,long_genoma);
% Seleccion de los padres
for i=1:2
% Selección del primer padre por torneo
   seleccion_torneo=randi(pop_size,n_torneo,1);
   valores_torneo=fitness_values(seleccion_torneo);
   [valor indice]=max(valores_torneo);
   padres(i,:)=poblacion(seleccion_torneo(indice),:);
end
end


function [hijo]= cruce_genetico(b,A,n,presupuesto,existentes,poblacion,long_genoma,pop_size,n_torneo,fitness_values)


% Cruce disperso
% Aleatoriamente se selecciona la mitad de genes de cada padre
% En lugar de generar un vector de binarios, se hace la seleccion de
% los genes dentro de un ciclo FOR para no almacenar en memoria el
% vector binario
hijo=zeros(1,long_genoma);
feasible=0;
while feasible~=1
   padres=seleccion_padres(long_genoma,pop_size,n_torneo,poblacion,fitness_values);
for j = 1:long_genoma
if(rand > 0.5)
hijo(j) = padres(1,j);
else
        hijo(j) = padres(2,j);
end
end
if factible(A,n,presupuesto,existentes,hijo,b)
break;
else
% Cruce de único punto
% Aleatoriamente se genera el pounto de corte.
punto_corte= ceil(rand * (long_genoma- 1));
% make one child
hijo= [padres(1,1:punto_corte),padres(1,( punto_corte+ 1 ):end )];
if factible(A,n,presupuesto,existentes,hijo,b)
break;
else
% Cruce aritmetico
        hijo=round((padres(1,:)+padres(2,:))/2);
if factible(A,n,presupuesto,existentes,hijo,b)
break;
else
continue;
end
end
end
end
end
```

```
function mutaciones = mutacion_factible(b,A,n,presupuesto,existentes,long_genoma,UB
,LB,n_generaciones,max_generaciones,n_mutacion,poblacion,escala,contraccion)
escala= escala- contraccion* contraccion* n_generaciones/max_generaciones;
escala= escala* (UB - LB);
mutaciones = zeros(n_mutacion,long_genoma);

j=1;
while j<=n_mutacion
% Se genera aleatoriamente un vector con la magnitud de las mutaciones
   mutacion= poblacion(j,:) + round(escala.* randn(1,long_genoma));
   mutacion(mutacion<LB)=LB(mutacion<LB);
   mutacion(mutacion>UB)=UB(mutacion>UB);
% Se verifica si la solucion tiene conectividad y no excede el presupuesto
if factible(A,n,presupuesto,existentes,mutacion,b)
      mutaciones(j,:)=mutacion;
      j=j+1;
end
end

% mutaciones = zeros(n_mutacion,long_genoma);
% j=1;
% while j<=n_mutacion
%    mutacion=poblacion(j,:);
%    posicion_mutacion=randi(long_genoma);
%    mutacion(posicion_mutacion)=randi([LB(posicion_mutacion),UB(posicion_mutacion)]);
%    if factible(A,n,presupuesto,existentes,mutacion,b)
%       mutaciones(j,:)=mutacion;
%       j=j+1;
%    end
% end

end

function [feasible]=factible(A,n,presupuesto,existentes,hijo,b)

% Funcion que verifica la legalidad y factibilidad de VECTOR

if hijo*A'<=presupuesto
% Si cumple la condición, se inspecciona la conectividad de la red
   red=zeros(n);
   red(b)=hijo;
   red=red+red';
   red_aumentada=existentes+red;
   feasible=conectividad(red_aumentada,n);
else
   feasible=false;
end
end
```

Apéndice 12

Confiabilidad de las mejores soluciones encontradas por los tres métodos – Tabla 1

Densidad	Conf. Prom. Arcos	Nodos	Método		
			1	2	3
0,333333333	0,05	15	0,3225	0,096363	0,08736912
0,333333333	0,05	30	0,7026	0,48633972	0,4883471
0,333333333	0,05	45	0,9574	0,93269908	0,95003984
0,333333333	0,45	15	0,9997	0,9997	1
0,333333333	0,45	30	1	1	1
0,333333333	0,45	45	1	1	1
0,333333333	0,85	15	1	1	1
0,333333333	0,85	30	1	1	1
0,333333333	0,85	45	1	1	1
0,666666667	0,05	15	0,8913	0,79174179	0,78046038
0,666666667	0,05	30	0,9923	0,99091078	0,99680252
0,666666667	0,05	45	0,9999	0,9999	1
0,666666667	0,45	15	1	1	1
0,666666667	0,45	30	1	1	1
0,666666667	0,45	45	1	1	1
0,666666667	0,85	15	1	1	1
0,666666667	0,85	30	1	1	1
0,666666667	0,85	45	1	1	1
0,333333333	0,05	15	0,1079	0,0112216	0,0091
0,333333333	0,05	30	0,6966	0,49730274	0,49787386
0,333333333	0,05	45	0,9554	0,93323472	0,9479844
0,333333333	0,45	15	0,9987	0,9987	0,9999
0,333333333	0,45	30	1	1	1
0,333333333	0,45	45	1	1	1
0,333333333	0,85	15	1	1	1
0,333333333	0,85	30	1	1	1
0,333333333	0,85	45	1	1	1
0,666666667	0,05	15	0,7771	0,5820479	0,5604018
0,666666667	0,05	30	0,9951	0,99370686	0,99700224
0,666666667	0,05	45	0,9982	0,9982	1
0,666666667	0,45	15	1	1	1
0,666666667	0,45	30	1	1	1
0,666666667	0,45	45	1	1	1
0,666666667	0,85	15	1	1	1
0,666666667	0,85	30	1	1	1

Confiabilidad de las mejores soluciones encontradas por los tres métodos – Tabla 2

Densidad	Conf. Prom. Arcos	Nodos	Método		
			1	2	3
0,666666667	0,85	45	1	1	1
0,166666667	0,05	15	0,00002197	0,00191139	0,0010701
0,166666667	0,05	30	0,0619	0,00359639	0,00346857
0,166666667	0,05	45	0,3254	0,1000605	0,09320325
0,166666667	0,45	15	0,9544	0,91212008	0,90896627
0,166666667	0,45	30	0,9955	0,99420585	0,99720195
0,166666667	0,45	45	0,9996	0,9996	0,9999
0,166666667	0,85	15	0,9999	0,9999	1
0,166666667	0,85	30	0,8619	0,8619	1
0,166666667	0,85	45	1	1	1
0,083333333	0,05	15	0,00001609	0,01346733	0,00597618
0,083333333	0,05	30	5,992E-07	0,1375164	0,08723295
0,083333333	0,05	45	9,417E-07	0,8239875	0,205975
0,083333333	0,45	15	0,4182	0,12437268	0,10703426
0,083333333	0,45	30	0,4383	0,19333413	0,19280481
0,083333333	0,45	45	0,9252	0,8599734	0,86341255
0,083333333	0,85	15	0,8133	0,81313734	0,99960004
0,083333333	0,85	30	0,9999	0,9999	1
0,083333333	0,85	45	0,9998	0,9998	1
0,166666667	0,05	15	0,0002881	0,0074906	0,0064428
0,166666667	0,05	30	0,0184	0,00032016	0,00022446
0,166666667	0,05	45	0,147	0,0215796	0,02279804
0,166666667	0,45	15	0,82	0,68224	0,6889792
0,166666667	0,45	30	0,9964	0,99361008	0,9942084
0,166666667	0,45	45	0,9999	0,9999	1
0,166666667	0,85	15	0,9987	0,9987	1
0,166666667	0,85	30	1	1	1
0,166666667	0,85	45	1	1	1
0,083333333	0,05	15	0,00003594	0,06203244	0,0486732
0,083333333	0,05	30	0	0	0
0,083333333	0,05	45	0,00005875	0,31302	0,11924064
0,083333333	0,45	15	0,6728	0,45743672	0,45492109
0,083333333	0,45	30	0,8579	0,73024448	0,72752064
0,083333333	0,45	45	0,984	0,973176	0,9770331
0,083333333	0,85	15	0,7438	0,55361034	0,55390806
0,083333333	0,85	30	0,9997	0,9997	1
0,083333333	0,85	45	0,9925	0,99240075	0,99980001

Confiabilidad de las mejores soluciones encontradas por los tres métodos – Tabla 3

Densidad	Conf. Prom. Arcos	Nodos	Método		
			1	2	3
0,041666667	0,05	15	7,208E-06	0,00425272	0,0035105
0,041666667	0,05	30	0	0	0,0305046
0,041666667	0,05	45	0	0	0,0192516
0,041666667	0,45	15	0,4495	0,21319785	0,19626534
0,041666667	0,45	30	0,4916	0,242113	0,23526725
0,041666667	0,45	45	0,4783	0,22212252	0,21808224
0,041666667	0,85	15	0,8223	0,71022051	0,73984542
0,041666667	0,85	30	0,9931	0,98684347	0,98743969
0,041666667	0,85	45	0,849	0,7220745	0,72335025
0,020833333	0,05	15	0	0	0,00317952
0,020833333	0,05	30	0	0	0,00045605
0,020833333	0,05	45	0	0	0
0,020833333	0,45	15	0,0278	0,000695	0,000675
0,020833333	0,45	30	0,023	0,0004186	0,00046228
0,020833333	0,45	45	0,0023	0,00000299	0,00000312
0,020833333	0,85	15	0,9221	0,90107612	0,92189048
0,020833333	0,85	30	0,9477	0,92732445	0,95687515
0,020833333	0,85	45	0,3804	0,2428854	0,4514195
0,041666667	0,05	15	2,167E-07	0,004334	0,002098
0,041666667	0,05	30	0	0	0,00458496
0,041666667	0,05	45	0	0	0
0,041666667	0,45	15	0,2114	0,043337	0,0428245
0,041666667	0,45	30	0,4078	0,15402606	0,147303
0,041666667	0,45	45	0,5432	0,29408848	0,2953337
0,041666667	0,85	15	0,6612	0,48816396	0,63360906
0,041666667	0,85	30	0,9773	0,95980633	0,96452041
0,041666667	0,85	45	0,9987	0,99810078	0,99870042
0,020833333	0,05	15	0	0	0,07373495
0,020833333	0,05	30	0	0	0
0,020833333	0,05	45	0	0	0
0,020833333	0,45	15	0,0202	0,00040804	0,00041006
0,020833333	0,45	30	0,0068	0,00000952	0,0000084
0,020833333	0,45	45	0,0591	0,00312048	0,00297264
0,020833333	0,85	15	0,854	0,7248752	0,72470544
0,020833333	0,85	30	0,8036	0,66104136	0,6761772

Apéndice 13

Tiempo de ejecución de los tres métodos – Tabla 1

Densidad	Conf. Prom. Arcos	Nodos	Método 1	2	3
0.333333333	0.05	15	1831.8	90	48.9
0.333333333	0.05	30	12484	1531	73
0.333333333	0.05	45	61365	4242	308
0.333333333	0.45	15	194.3209	44.5146	25.5715
0.333333333	0.45	30	568.5183	144.3508	39.2363
0.333333333	0.45	45	2146.8	405.8	67.5
0.333333333	0.85	15	105.8528	44.3368	23.3489
0.333333333	0.85	30	487.5606	140.289	37.7996
0.333333333	0.85	45	1898	345.9	63
0.666666667	0.05	15	4588.2	257.9	55.6
0.666666667	0.05	30	2114.9	139.1	84.1
0.666666667	0.05	45	6947.5	337.8	85.6
0.666666667	0.45	15	142.9942	44.7411	23.5473
0.666666667	0.45	30	909.0613	141.4581	38.1268
0.666666667	0.45	45	3265.5	345.3	62.9
0.666666667	0.85	15	142.393	44.4557	23.3427
0.666666667	0.85	30	874.014	139.3811	37.5314
0.666666667	0.85	45	3622	389.6	63.9
0.333333333	0.05	15	1419.2	29.1	29.7
0.333333333	0.05	30	11736	446	82
0.333333333	0.05	45	48601	319	302
0.333333333	0.45	15	172.2121	43.7528	28.4645
0.333333333	0.45	30	503.3578	142.2059	38.1863
0.333333333	0.45	45	1899.1	351.2	63.8
0.333333333	0.85	15	134.5432	43.2705	22.6318
0.333333333	0.85	30	540.5315	139.6733	37.6849
0.333333333	0.85	45	1964.7	352.5	64.1
0.666666667	0.05	15	3601.4	197	21
0.666666667	0.05	30	2020.9	139.7	59.8
0.666666667	0.05	45	3352.5	344	87.8
0.666666667	0.45	15	131.6293	44.1999	23.1385
0.666666667	0.45	30	964.8599	142.8432	38.3077
0.666666667	0.45	45	3909.3	404.3	66.1
0.666666667	0.85	15	125.9559	43.9742	22.8261
0.666666667	0.85	30	912.1166	141.2043	37.5799
0.666666667	0.85	45	3857.6	397.3	64.7

Tiempo de ejecución de los tres métodos – Tabla 2

Densidad	Conf. Prom. Arcos	Nodos	Método		
			1	2	3
0.166666667	0.05	15	43.5058	24.3623	11.6953
0.166666667	0.05	30	433.3732	471.6856	62.4573
0.166666667	0.05	45	21500	2205	62
0.166666667	0.45	15	755.3291	65.6713	146.0249
0.166666667	0.45	30	540.4157	133.6101	81.5999
0.166666667	0.45	45	2243.2	327.6	83.1
0.166666667	0.85	15	108.8247	42.6066	28.555
0.166666667	0.85	30	371.2396	134.9644	36.2057
0.166666667	0.85	45	1100.5	329.2	59.9
0.083333333	0.05	15	47.8534	22.9901	11.071
0.083333333	0.05	30	199.3729	61.8057	16.5747
0.083333333	0.05	45	544.181	134.0594	23.8196
0.083333333	0.45	15	81.4333	133.4686	22.0337
0.083333333	0.45	30	2629.3	108.7	105.6
0.083333333	0.45	45	13532	303	56
0.083333333	0.85	15	31.8119	37.9392	91.9555
0.083333333	0.85	30	382.4139	126.2878	36.741
0.083333333	0.85	45	1367.2	330.2	59.7
0.166666667	0.05	15	55.517	24.9154	11.8143
0.166666667	0.05	30	429.6212	74.3127	45.1703
0.166666667	0.05	45	18176	1343	88
0.166666667	0.45	15	220.5414	31.5979	196.5051
0.166666667	0.45	30	607.8404	136.444	55.9352
0.166666667	0.45	45	1236.6	342.1	62.2
0.166666667	0.85	15	66.8175	41.7709	24.8497
0.166666667	0.85	30	554.0542	137.4611	36.724
0.166666667	0.85	45	1140.7	331.2	60
0.083333333	0.05	15	41.3769	23.0806	11.2311
0.083333333	0.05	30	100.394	58.9402	15.1986
0.083333333	0.05	45	462.8931	139.8569	24.6457
0.083333333	0.45	15	69.2491	101.485	313.8901
0.083333333	0.45	30	3154.1	1115.8	1385.2
0.083333333	0.45	45	11006	310	92
0.083333333	0.85	15	4.1638	3.4211	5.1027
0.083333333	0.85	30	298.0422	116.4	31.774
0.083333333	0.85	45	1141.4	320.8	58.5

Tiempo de ejecución de los tres métodos – Tabla 3

Densidad	Conf. Prom. Arcos	Nodos	Método		
			1	2	3
0.041666667	0.05	15	32.6389	22.7525	0.3776
0.041666667	0.05	30	85.124	52.01	12.6151
0.041666667	0.05	45	2.3608	94.0144	15.9361
0.041666667	0.45	15	0.3665	0.5986	0.3899
0.041666667	0.45	30	1411.2	307.6	53.3
0.041666667	0.45	45	5288.8	1617.2	107.3
0.041666667	0.85	15	0.1463	0.0765	0.1841
0.041666667	0.85	30	184.8239	86.8633	37.6118
0.041666667	0.85	45	379.6559	241.5054	41.7224
0.020833333	0.05	15	0.0528	0.1216	0.0595
0.020833333	0.05	30	1.3601	41.1221	4.4927
0.020833333	0.05	45	2.0547	76.506	11.8055
0.020833333	0.45	15	0.1507	0.0798	0.0318
0.020833333	0.45	30	67.8529	72.2563	18.751
0.020833333	0.45	45	97.1339	103.2958	16.4574
0.020833333	0.85	15	0.1303	0.0991	0.0916
0.020833333	0.85	30	1199.7	81	1208.2
0.020833333	0.85	45	251.1202	757.8283	109.7662
0.041666667	0.05	15	0.1592	0.0913	0.05
0.041666667	0.05	30	1.7734	53.3312	12.718
0.041666667	0.05	45	2.6955	113.7367	19.5849
0.041666667	0.45	15	91.12	27.1408	11.7147
0.041666667	0.45	30	1756.2	343.3	53.2
0.041666667	0.45	45	4223.9	2494	116.6
0.041666667	0.85	15	0.1673	0.0767	0.1188
0.041666667	0.85	30	174.1	486	1349
0.041666667	0.85	45	559.1904	255.3192	76.7771
0.020833333	0.05	15	0.0523	0.1204	0.0387
0.020833333	0.05	30	0.1321	0.1089	0.1937
0.020833333	0.05	45	19.6368	84.2282	13.9957
0.020833333	0.45	15	0.1339	0.0829	0.0494
0.020833333	0.45	30	82.8616	75.6272	30.3483
0.020833333	0.45	45	186.8	1293.9	80.6
0.020833333	0.85	15	0.2637	0.0738	0.089
0.020833333	0.85	30	270.9307	81.9879	286.1376
0.020833333	0.85	45	0.7724	0.1766	0.3792